薛定宇教授大讲堂（卷Ⅲ）

MATLAB
线性代数运算

薛定宇◎著
Xue Dingyu

Professor Xue Dingyu's Lecture Hall（Volume Ⅲ）
MATLAB Linear Algebra Operation

清華大學出版社
北京

内 容 简 介

本书按照线性代数教材的编排方式，系统论述了基于 MATLAB 语言编程的方法来实现线性代数问题的求解。全书内容包括矩阵的输入方法、矩阵基本分析方法、矩阵基本变换与分解方法、矩阵方程的求解方法与矩阵任意函数的计算方法等。此外，书中还介绍了线性代数的诸多应用问题的建模与求解方法。

本书可以作为高等学校理工科各类专业的本科生与研究生学习计算机数学语言（MATLAB）的教材，也可以作为一般读者学习线性代数与矩阵分析的辅助教材——从另一个角度认识线性代数问题的求解方法，并可以作为查询线性代数与矩阵数学问题求解方法的工具书。

图书在版编目（CIP）数据

薛定宇教授大讲堂(卷Ⅲ): MATLAB 线性代数运算/薛定宇著. —北京：清华大学出版社，2019
ISBN 978-7-302-51870-9

Ⅰ. ①薛… Ⅱ. ①薛… Ⅲ. ①MATLAB 软件-应用-线性代数-高等学校-教学参考资料 Ⅳ. ①O151.2-39

中国版本图书馆 CIP 数据核字(2018)第 285092 号

责任编辑：盛东亮
封面设计：李召霞
责任校对：李建庄
责任印制：
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：186mm×240mm　　**印　张**：15.75　　**字　　数**：310 千字
版　　次：2019 年 7 月第 1 版　　**印　　次**：2019 年 7 月第 1 次印刷
定　　价：00.00 元

产品编号：078636-01

前　言

PREFACE

科学运算问题是每个理工科学生和科技工作者在课程学习、科学研究与工程实践中常常会遇到的问题，不容回避。对于非纯数学专业的学生和研究者而言，从底层全面学习相关数学问题的求解方法并非一件简单的事情，也不易得出复杂问题的解。所以，利用当前最先进的计算机工具，高效、准确、创造性地求解科学运算问题是一种行之有效的方法，尤其能够满足理工科人士的需求。

作者曾试图在同一部著作中叙述各个数学分支典型问题的直接求解方法，通过清华大学出版社出版了《高等应用数学问题的MATLAB求解》。该书从2004年出版之后多次重印再版，并于2018年出版了第4版，还配套发布了全新的MOOC课程[①]，一直受到广泛的关注与欢迎。首次MOOC开课的选课人数接近14000人，教材内容也被数万篇期刊文章和学位论文引用。

从作者首次使用MATLAB语言算起，已经30余年了，通过相关领域的研究、思考与一线教学实践，积累了大量的实践经验资料。这些不可能在一部著作中全部介绍，所以与清华大学出版社策划与编写了这套“薛定宇教授大讲堂”系列著作，系统深入地介绍基于MATLAB语言与工具的科学运算问题的求解方法。

本系列著作不是原来版本的简单改版，通过十余年的经验和资料积累，全面贯穿“再认识”的思想写作此书，深度融合科学运算数学知识与基于MATLAB的直接求解方法与技巧，力图更好地诠释计算机工具在每个数学分支的作用，帮助读者以不同的思维与视角了解工程数学问题的求解方法，创造性地得出问题的解。

本系列著作卷I可以作为学习MATLAB入门知识的教材与参考书，也为读者深入学习与熟练掌握MATLAB语言编程技巧，深度理解科学运算领域MATLAB的应用奠定一个坚实的基础。后续每一卷试图对应一个数学专题或一门数学课程进行展开。整套系列著作的写作贯穿“计算思维”的思想，深度探讨该数学专题的问题求解方法。本系列著作既适合学完相应的数学课程之后，深入学习利用计算机

① MOOC网址：https://www.icourse163.org/learn/NEU-1002660001

工具的科学运算问题求解方法与技巧，也可作为相应数学课程同步学习的伴侣，在学习相应课程理论知识的同时，侧重于学习基于计算机的数学问题求解方法，从另一个角度观察、审视数学课程所学的内容，扩大知识面，更好地学习、理解并实践相应的数学课程。

本书是系列著作的卷Ⅲ。本书试图以一个全新的角度，按照一般线性代数教程的方式介绍线性代数问题的求解，侧重利用MATLAB语言直接求解矩阵运算与线性代数的问题。首先介绍矩阵的输入方法，然后介绍矩阵基本分析方法、矩阵基本变换与分解方法，并介绍矩阵方程的求解方法与矩阵任意函数的计算方法等。本书还介绍了线性代数的诸多应用问题的建模与求解方法。

值此系列著作付梓之际，衷心感谢相濡以沫的妻子杨军教授，她数十年如一日的无私关怀是我坚持研究、教学与写作工作的巨大动力。

薛定宇

2019年5月

目　录

CONTENTS

第1章

线性代数简介

1.1 矩阵与线性方程组

线性代数的研究起源于对线性方程组的求解。线性方程组是科学研究与工程实践中应用最广泛的数学模型，在实际应用中还可能建立更复杂的线性代数方程。为了研究方便，引入矩阵描述代数方程组。本节将给出几个简单的例子，演示数据表格的矩阵表示方法，并说明线性方程组的重要性。

1.1.1 表格的矩阵表示

在人们的日常生活与科学研究中，经常会遇到各种各样的数据表格。如何有效地表示这些表格呢？表格在数学中和计算机上可能有各种各样的表示方法，矩阵是数据表格最有效的表示方法之一。

定义 1-1 矩阵的数学形式为

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & a_{23} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \tag{1-1-1}$$

矩阵是线性代数领域重要的数学单元，下面通过例子演示用矩阵表示表格的具体方法。

例 1-1 彩色图像的颜色在计算机上有多种表示方法。其中，三原色法是一种重要的颜色表示方法，一种颜色可以理解成由红(R)、绿(G)、蓝(B)三个颜色分量的不同组合构成。常用八种颜色的RGB三原色分量如表1-1所示，试用矩阵表示该表格。

表 1-1 常用颜色的RGB分量

三原色分量	黑色	蓝色	绿色	青色	红色	品红	黄色	白色
红	0	0	0	0	255	255	255	255
绿	0	0	255	255	0	0	255	255
蓝	0	255	0	255	0	255	0	255

解 如果矩阵的行用于表示三原色，各列分别表示黑色、蓝色、……、白色，则可以用一个3×8的矩阵表示整个表格，这个矩阵的元素排列与表格的数据排列完全一致，即

$$\boldsymbol{A}=\begin{bmatrix}0&0&0&0&255&255&255&255\\0&0&255&255&0&0&255&255\\0&255&0&255&0&255&0&255\end{bmatrix}$$

有了矩阵的数学表达式，用下面的语句将其直接输入MATLAB的工作空间，就可以通过相应的命令对其进行运算了。

```
>> A=[0 0 0 0 255 255 255 255; 0 0 255 255 0 0 255 255
      0 255 0 255 0 255 0 255];
```

从给出的表格可见，品红色是矩阵的第六列，所以可以由下面的命令提取出品红色的红绿蓝颜色分量

```
>> c=A(:,6)
```

例1-2 八大行星的一些参数由表1-2中给出。其中，相对参数都是由地球参数换算得到的，半长轴的单位为AU（Astronomical Unit，天文单位，为$149597870700\,\mathrm{m}\approx 1.5\times 10^{11}\,\mathrm{m}$，地球到太阳的平均距离），自转周期的单位为天。试用矩阵表示这个表格。

表1-2 八大行星的一些参数

名称	相对直径	相对质量	半长轴	相对轨道周期	离心率	自转周期	卫星个数	行星环
水星	0.382	0.06	0.39	0.24	0.206	58.64	0	无
金星	0.949	0.82	0.72	0.62	0.007	243.02	0	无
地球	1	1	1	1	0.017	1	1	无
火星	0.532	0.11	1.52	1.88	0.093	1.03	2	无
木星	11.209	317.8	5.20	11.86	0.048	0.41	69	有
土星	9.449	95.2	9.54	29.46	0.054	0.43	62	有
天王星	4.007	14.6	19.22	84.01	0.047	0.72	27	有
海王星	3.883	17.2	30.06	164.8	0.009	0.67	14	有

解 观察表1-2可以发现，表格中大部分元素都是数值。除了数值之外还有表头，表格第一列为“名称”。此外，最后一列数据的内容为“无”或“有”。对最后一列进行变换，令“无”为0、“有”为1，则最后一列也是数据。如果只关心这个表格中的数据，不妨用矩阵更简洁地表示这个表格，即

$$\boldsymbol{A}=\begin{bmatrix}0.382&0.06&0.39&0.24&0.206&58.64&0&0\\0.949&0.82&0.72&0.62&0.007&-243.02&0&0\\1&1&1&1&0.017&1&1&0\\0.532&0.11&1.52&1.88&0.093&1.03&2&0\\11.209&317.8&5.2&11.86&0.048&0.41&69&1\\9.449&95.2&9.54&29.46&0.054&0.43&62&1\\4.007&14.6&19.22&84.01&0.047&-0.72&27&1\\3.883&17.2&30.06&164.8&0.009&0.67&14&1\end{bmatrix}$$

有了矩阵的数学形式，则可以用下面的MATLAB语句进行输入。

```
>> A=[0.382,0.06,0.39,0.24,0.206,58.64,0,0;
      0.949,0.82,0.72,0.62,0.007,-243.02,0,0;
      1,1,1,1,0.017,1,1,0;
      0.532,0.11,1.52,1.88,0.093,1.03,2,0;
      11.209,317.8,5.2,11.86,0.048,0.41,69,1;
      9.449,95.2,9.54,29.46,0.054,0.43,62,1;
      4.007,14.6,19.22,84.01,0.047,-0.72,27,1;
      3.883,17.2,30.06,164.8,0.009,0.67,14,1];
```

本丛书卷 I 中用到了这个例子，使用 MATLAB 下的 table 数据结构表示表 1-2。下面给出相应的 MATLAB 命令。

```
>> name=str2mat('水星','金星','地球','火星','木星','土星',...
                '天王星','海王星');
   diameter=[0.382;0.949;1;0.532;11.209;9.449;4.007;3.883];
   mass=[0.06; 0.82; 1; 0.11; 317.8; 95.2; 14.6; 17.2];
   axis=[0.39; 0.72; 1; 1.52; 5.2; 9.54; 19.22; 30.06];
   period=[0.24; 0.62; 1; 1.88; 11.86; 29.46; 84.01; 164.8];
   eccentricity=[0.206; 0.007; 0.017; 0.093; 0.048; ...
        0.054; 0.047; 0.009];
   rotation=[58.64;-243.02;1;1.03;0.41;0.43;-0.72;0.67];
   moon=[0; 0; 1; 2; 69; 62; 27; 14];
   ring={'无';'无';'无';'无';'有';'有';'有';'有'};
   planet=table(name,diameter,mass,axis,period,eccentricity,...
                rotation,moon,ring)
```

例 1-3　例 1-2 给出的相对数据是地球数据的倍数。已知地球的质量为 5.965×10^{24} kg，试求出其他行星的质量，例如木星的质量。

解　从矩阵的存储看，“相对质量”是矩阵的第二列，第二列的全部元素可以由 $\boldsymbol{A}$(:,2) 命令提取。木星是第五行，所以可以用下面的命令计算出各个行星的实际质量，提取第五个元素则为木星的质量，为 1.8957×10^{27} kg。

```
>> M0=5.965e24; M=A(:,2)*M0; M(5)
```

1.1.2　线性方程组的建立与求解

线性代数的研究起源于线性方程组的列写与求解，本节给出几个例子演示实际问题的线性方程组建模方法。

例 1-4　公元 4 – 5 世纪的中国古代著名的数学著作《孙子算经》曾给出了鸡兔同笼问题：“今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？”

解　古典数学著作中有各种各样的方法求解鸡兔同笼问题。如果引入代数方程的思维，则假设鸡的个数为 x_1，兔的个数为 x_2，可以列出下面的线性代数方程。

$$\begin{cases} x_1 + x_2 = 35 \\ 2x_1 + 4x_2 = 94 \end{cases}$$

如果引入矩阵的概念，则可以将线性代数方程写成矩阵形式，即

$$\begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 35 \\ 94 \end{bmatrix}$$

记 $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 35 \\ 94 \end{bmatrix}$，则可以写出线性代数方程的标准形式：

$$\boldsymbol{Ax} = \boldsymbol{B} \tag{1-1-2}$$

线性方程在MATLAB下的求解语句为 $\boldsymbol{x}$=$\boldsymbol{A}$\$\boldsymbol{B}$，所以由MATLAB命令求解方程，得出 $\boldsymbol{x} = [23, 12]^{\mathrm{T}}$。方程解的物理含义是，鸡有23个，兔有12个。

```
>> A=[1 1; 2 4]; B=[35; 94]; x=A\B
```

此外，还可以由符号运算中解方程的方法（不限于线性代数方程组）求解鸡兔同笼问题，其结果与前面得出的完全一致。

```
>> syms x y; [x y]=solve(x+y==35, 2*x+4*y==94)
```

注意：早期版本中，上面的语句可以使用字符串描述方程本身。但新版本不支持这种形式，应该采用符号表达式表示方程。

例 1-5　文献[1]给出了梁平衡问题的应用实例。假设一个梁系统的结构体如图1-1所示，每条线段表示一根梁，每一个圆圈表示一个连接点。假设所有斜线梁的倾斜角度都为45°，且连接点1的水平与垂直方向都固定，连接点8的垂直方向固定，在连接点2,5,6处增加负载。为使得整个架构平衡，试根据各个连接点的水平和垂直方向列出线性代数方程，写出其矩阵方程形式并试图得出方程的解。

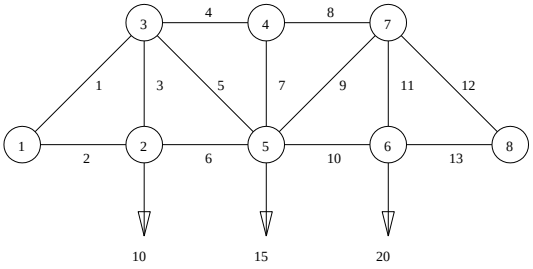


图 1-1　梁的平面结构

解　由于连接点1水平与垂直方向都固定了，所以无须为其列写水平方向与垂直方向的平衡方程。现在考虑连接点2，从水平方向看，该连接点受两个力的影响，一个是梁2施加的力，记作 f_2，另一个是梁6施加的力，记作 f_6。为使得水平方向平衡，显然需要满足 $f_2 = f_6$，或写作 $f_2 - f_6 = 0$。

再对连接点2的垂直方向进行受力分析，该连接点受梁3的力与外力10，所以相应的平衡方程为 $f_3 = 10$。

为列写方程方便，后面统一设置连接点左边和上面的力为正方向，否则为负方向。

现在分析连接点3处的水平平衡方程：在水平方向该点受f_1、f_4和f_5三个力的同时作用。其中，f_1与f_5是倾斜方向的力，由于倾斜角为45°，所以应该乘以$\cos 45° = 1/\sqrt{2}$。这样，记$\alpha = 1/\sqrt{2}$，则水平方向的平衡方程为$\alpha f_1 - f_4 - \alpha f_5 = 0$。再考虑垂直方向的平衡方程，不难看出$\alpha f_1 + f_3 + \alpha f_5 = 0$。

类似地，可以写出连接点4的水平方向平衡方程为$f_4 - f_8 = 0$，垂直方向平衡方程为$f_7 = 0$。

连接点5的平衡方程为$\alpha f_5 + f_6 - \alpha f_9 - f_{10} = 0$，$f_5 + f_7 + \alpha f_9 = 15$。

连接点6的水平、垂直平衡方程分别为$f_{10} - f_{13} = 0$和$f_{11} = 20$。

连接点7的水平、垂直平衡方程分别为$f_8 + \alpha f_9 - \alpha f_{12} = 0$和$\alpha f_9 + f_{11} + \alpha f_{12} = 0$。

连接点8，由于垂直方向固定，只能列出水平方向的平衡方程为$\alpha f_{12} + f_{13} = 0$。

上面总共列出了13个方程，写成矩阵形式即

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha & 0 & 0 & -1 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & 1 & 0 & 0 & -\alpha & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -\alpha & 0 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 1 & \alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \\ f_{11} \\ f_{12} \\ f_{13} \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 15 \\ 0 \\ 20 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

采用手工方式求解这样的方程是很困难的，所以应该设法将其送给计算机去求解。后面将介绍具体的求解方法，这里只给出一个演示。

```
>> alpha=1/sqrt(sym(2));
   A=[0  1  0  0  0  -1  0  0  0  0  0  0  0
      0  0  1  0  0  0  0  0  0  0  0  0  0
      alpha  0  0  -1  -alpha  0  0  0  0  0  0  0  0
      alpha  0  1  0  alpha  0  0  0  0  0  0  0  0
      0  0  0  1  0  0  0  -1  0  0  0  0  0
      0  0  0  0  0  0  1  0  0  0  0  0  0
      0  0  0  0  alpha  1  0  0  -alpha  -1  0  0  0
      0  0  0  0  1  0  1  0  alpha  0  0  0  0
      0  0  0  0  0  0  0  0  0  1  0  0  -1
      0  0  0  0  0  0  0  0  0  0  1  0  0
      0  0  0  0  0  0  0  1  -alpha  0  0  -alpha  0
      0  0  0  0  0  0  0  0  alpha  0  1  alpha  0
      0  0  0  0  0  0  0  0  0  0  0  alpha  1];
```

```
B=[0; 10; 0; 0; 0; 0; 0;15; 0; 20; 0; 0; 0];
f=simplify(A\B), save c1dat1 A B f alpha
```

由以上语句可以立即得出：$f_1=-15\sqrt{2}$，$f_2=45-10\sqrt{2}$，$f_3=10$，$f_4=-20$，$f_5=5\sqrt{2}$，$f_6=45-10\sqrt{2}$，$f_7=0$，$f_8=-20$，$f_9=15\sqrt{2}-10$，$f_{10}=35-5\sqrt{2}$，$f_{11}=20$，$f_{12}=10-35\sqrt{2}$，$f_{13}=35-5\sqrt{2}$。

在例1-5中，未知数的个数与方程的个数是相等的，得出的方程解是唯一的，这类方程又称为洽定方程（consistent equation）。在一些特殊情况下，还应该考虑未知数个数不同的方程。

例1-6　考虑例1-5中的问题。如果不固定连接点1与连接点8，则可以再建立三个方程，试写出其线性代数方程及其矩阵形式。

解　如果不固定连接点1和连接点8，则这样的结构不能悬空放置，需要在连接点1与连接点8处引入支撑力s_1和s_2，如图1-2所示。

如果连接点1不再固定，则可以写出其水平方向的平衡方程为$\alpha f_1+f_2=s_3$，垂直方向的平衡方程为$\alpha f_1=s_1$；如果不固定连接点8的垂直方向，则可以写出垂直方向的平衡方程为$\alpha f_{12}=s_2$。这样，原来的矩阵方程可以改写成

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\alpha & 0 & 0 & -1 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\alpha & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & \alpha & 1 & 0 & 0 & -\alpha & -1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -\alpha & 0 & 0 & -\alpha & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 1 & \alpha & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 1\\
\alpha & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0
\end{bmatrix}
\begin{bmatrix} f_1\\ f_2\\ f_3\\ f_4\\ f_5\\ f_6\\ f_7\\ f_8\\ f_9\\ f_{10}\\ f_{11}\\ f_{12}\\ f_{13} \end{bmatrix}
=
\begin{bmatrix} 0\\ 10\\ 0\\ 0\\ 0\\ 0\\ 0\\ 15\\ 0\\ 20\\ 0\\ 0\\ 0\\ s_3\\ s_1\\ s_2 \end{bmatrix}$$

由以下语句可以直接输入系数矩阵。

```
>> load c1dat1
   A=[A;
      alpha, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;
      alpha, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, alpha, 0];
```

由于得到的方程个数大于未知数的个数（$\boldsymbol{A}$是16×13的长方形矩阵），则该方程称

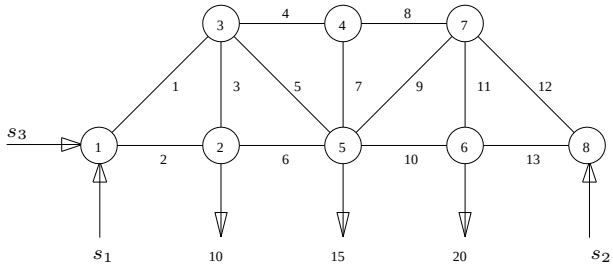


图 1-2　修改后的梁平面结构

为超定方程(over-determined equations)。如果将未知量 s_1、s_2 与 s_3 写入力向量 $\boldsymbol{f}$，则向量 $\boldsymbol{f}$ 有 16 个元素，相应的 $\boldsymbol{A}$ 矩阵则变成 16×16 的方阵，方程为恰定方程。

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\alpha & 0 & 0 & -1 & -\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\alpha & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & \alpha & 1 & 0 & 0 & -\alpha & -1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -\alpha & 0 & 0 & -\alpha & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 1 & \alpha & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 1 & 0 & 0 & 0\\
\alpha & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\
\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix} f_1\\ f_2\\ f_3\\ f_4\\ f_5\\ f_6\\ f_7\\ f_8\\ f_9\\ f_{10}\\ f_{11}\\ f_{12}\\ f_{13}\\ s_1\\ s_2\\ s_3 \end{bmatrix}
=
\begin{bmatrix} 0\\ 10\\ 0\\ 0\\ 0\\ 0\\ 0\\ 15\\ 0\\ 20\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0 \end{bmatrix}$$

有了恰定方程，则可以由以下语句直接求解方程的解析解。

```
>> A=[A, [zeros(13,3); [0 0 1; 1 0 0; 0 1 0]]];
   x=simplify(inv(A)*[B;0;0;0]), simplify(A*x-[B; 0;0;0])
```

求得结果为：$f_1=-15\sqrt{2}$，$f_2=45-10\sqrt{2}$，$f_3=10$，$f_4=-20$，$f_5=5\sqrt{2}$，$f_6=45-10\sqrt{2}$，$f_7=0$，$f_8=-20$，$f_9=15\sqrt{2}-10$，$f_{10}=35-5\sqrt{2}$，$f_{11}=20$，$f_{12}=10-35\sqrt{2}$，$f_{13}=35-5\sqrt{2}$，$s_1=15$，$s_2=35-5\sqrt{2}$，$s_3=-30+10\sqrt{2}$。

可以看出，在放开几个固定端之后，f_i 的值保持不变，而 s_i 的值可以由新方程解出。将方程的解代入原方程，可以看出满足是原方程的。

超定方程的解有两种可能，一是方程有无穷多解，另一种是方程无解(得出的方程是矛盾方程)。

与超定方程相对应的还有欠定(under-determined)方程，后续本书将讨论这些方程的求解方法。

1.2　线性代数发展简介

1.2.1　线性代数数学理论

线性代数学科的起源是在求解线性方程时引入行列式的概念。德国数学家Gottfried Leibniz（1646−1716，图1-3（a））在1693年就研究过行列式的问题。1750年，瑞士数学家Gabriel Cramer（1704−1752，图1-3（b））利用行列式的概念给出了线性方程组的显式解法，该方法现在称为Cramer法则。后来，德国数学家Johann Gauss（1777−1855，图1-3（c））提出了求解线性方程组的方法，被后人称为Gauss消去法。


（a）Gottfried Leibniz


（b）Gabriel Cramer


（c）Johann Gauss

图 1-3　Leibniz、Cramer与Gauss画像

注：图像均来源于维基百科

对矩阵代数的系统研究最早出现于18世纪中叶的欧洲。1844年，德国数学家Hermann Grassmann（1809−1877，图1-4（a））出版了名为*Die Ausdehnungslehre von*（扩展的理论，在数学史上称为A1，1862年出版了A2）的著作[2]，创建了线性代数这一新的数学分支，将线性代数从几何学中独立出来，并引入了线性空间的雏形。1848年，英国数学家James Sylvester（1817−1897，图1-4（b））创造了“matrix”一词。而在研究矩阵变换时，英国数学家Arthur Cayley（1821−1895，图1-4（c））使用了一个字母表示整个矩阵，定义了矩阵相乘的方法，并给出了逆矩阵的概念。

1882年，奥斯曼帝国数学家Hüseyin Tevfik Paşha将军（1832−1901，图1-5（a））写了一部题为“*Linear Algebra*”（线性代数）的著作。1888年，意大利数学家Giuseppe Peano（1858−1932，图1-5（b））引入了线性空间的更精确的定义。1900年，出现了有限维向量空间的线性变换理论，并出现了线性代数的其他分支，拓展了线性代数这一分支的新应用成果。


（a）Hermann Grassmann


（b）James Sylvester

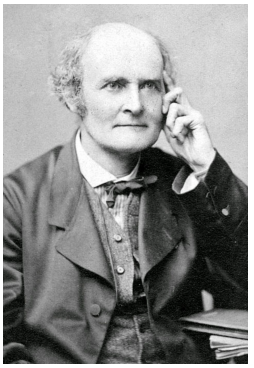
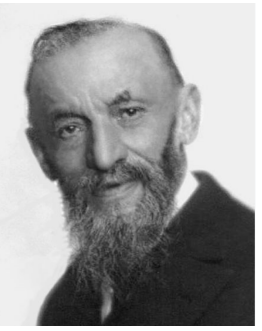

（c）Arthur Cayley

图 1-4　Grassmann、Sylvester与Cayley画像

注：图像均来源于维基百科

哈佛大学Wassily Leontief（1906—1999，图1-5（c））在经济学领域建立了投入产出分析理论，用来分析一个经济部门的变化如何影响其他部门。1949年底，他利用美国劳动统计局的25万条数据将美国经济分成500个部门，建立了500个未知数的线性方程。由于该方程当时无法求解，所以他将问题简化成42个方程，并花费几个月的时间编程并输入数据，在当时的计算机上花了56小时得到了方程的解[3]。Leontief开创了经济学的数学建模，并获得了1973年度诺贝尔经济学奖。随着计算机硬件与软件的发展，投入产出分析可以处理大规模矩阵方程问题，其迭代算法影响了1998年出现的谷歌佩奇排名（PageRank，又称网页排名），可以用于超大规模的矩阵运算。


（a）Hüseyin Paşha


（b）Giuseppe Peano

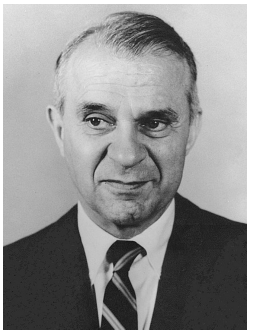

（c）Wassily Leontief

图 1-5　Paşha、Peano与Leontief照片

注：图像均来源于维基百科

在量子力学、狭义相对论、统计学、控制科学等领域，由于矩阵的大量使用，使线性代数理论从纯数学研究扩展到各个应用领域，而计算机的发展催生了Gauss

消去法、矩阵分解领域的高效算法，使得线性代数成为系统建模与仿真领域最基础的工具。

1.2.2　数值线性代数

早期线性代数发展过程中出现了一些线性代数问题的计算方法，这些计算方法可以认为是数值线性代数的雏形。数值线性代数是在数字计算机出现之后自然而然发展起来的线性代数分支，数值线性代数是线性代数与矩阵理论与应用发展巨大的推动力。

1950年前后，英国国家物理实验室的James Wilkinson（1919−1986，图1-6（a））开始在最早的计算机上开始了数值分析的研究，其学术著作包括文献[4]～[5]。与此同时，美国数值分析研究院的George Forsythe（1917−1972，图1-6（b））等也开展了数值分析，特别是数值线性代数方面的研究[6,7]。

美国数学家、计算机专家Cleve Moler（1939−，图1-6（c））等开发了当时国际最先进的数值线性代数软件包EISPACK与LINPACK等。Moler于1979年正式发布了他编写的MATLAB语言（取名自MATrix LABoratory，矩阵实验室），并与Jack Little等于1984年建立了MathWorks公司，致力于MATLAB语言的开发与应用研究。MATLAB语言已经成为很多领域的首选计算机语言，并影响了很多其他语言，对数值线性代数的研究与工程应用起了重要的作用。

（a）James Wilkinson

（b）George Forsythe

（c）Cleve Moler

图 1-6　Wilkinson、Forsythe与Moler照片

注：图像均来源于网络

2012年，Cleve Moler应邀在中国多所大学演讲，回顾数值分析发展的历史与MATLAB出现的背景。作者为其在同济大学演讲的视频配了英文字幕，建议读者观看该视频，领略大师的风采。

视频的百度网盘地址：`https://pan.baidu.com/s/1pNkhcnx`。

优酷网站：`http://v.youku.com/v_show/id_XNDc0NTM4NzQw.html`。

本书将系统地介绍线性代数，并侧重介绍基于MATLAB的线性代数问题求解方法。

第2章首先给出矩阵的基本概念，并介绍将一般矩阵输入MATLAB工作空间的方法，还介绍单位矩阵、随机矩阵、对角矩阵、Hankel矩阵等特殊矩阵的输入方法与任意符号矩阵、矩阵函数的输入方法。本章还介绍稀疏矩阵的输入方法、存储方式与转换方法，并介绍稀疏矩阵的非零元素的提取与图形表示方法。此外，本章介绍矩阵的基本运算方法，包括简单的代数运算、复数矩阵的处理以及矩阵的微积分运算等。

第3章侧重于介绍矩阵分析的基本内容，包括行列式、矩阵的迹、矩阵的线性相关与秩、向量范数与矩阵范数等，并介绍向量空间等概念。另外，本章还介绍逆矩阵的概念与求解方法以及行阶梯标准型变换问题，并介绍广义逆矩阵的计算方法。本章还介绍矩阵特征多项式与特征值的概念与计算方法，并给出矩阵多项式的概念与计算方法。

第4章侧重于介绍矩阵变换与分解。首先给出矩阵相似变换的概念与性质，并给出正交矩阵的概念。然后介绍初等行变换方法与规则，并在此基础上给出基于初等行变换的矩阵求逆方法与主元素的概念。另外，介绍线性代数方程的Gauss消去法、矩阵的三角分解方法、对称矩阵的Cholesky分解方法等，并给出正定矩阵与正规矩阵的概念与判定方法。本章还介绍将一般矩阵变换为相伴矩阵、对角矩阵及Jordan矩阵的方法，并给出矩阵的奇异值分解、矩阵条件数的概念与求解方法。此外，本章还介绍Givens变换与Householder变换的方法。

第5章侧重于介绍各种矩阵方程的计算机求解方法。首先介绍线性代数方程组的数值与解析求解方法，对线性代数方程进行分类，分别求解代数方程的唯一解、无穷解与最小二乘解。本章还介绍各种Lyapunov方程、Sylvester方程等的数值与解析解求解方法，并介绍Riccati方程的数值求解方法，还试图求解出不同类型的Riccati方程全部的根，并给出一般非线性矩阵方程的通用求解MATLAB函数，适合于求解任意复杂的非线性矩阵方程。本章还介绍多项式Diophantine方程的解析解方法，并介绍Bézout恒等式的求解方法。

第6章侧重于介绍矩阵函数的数值与解析运算。首先给出矩阵函数的定义，介绍最常用的矩阵指数运算，并介绍矩阵的对数函数、平方根函数和三角函数运算。本章还给出了矩阵任意函数的计算方法与MATLAB实现，并介绍了矩阵任意乘方$\boldsymbol{A}^k$与$k^{\boldsymbol{A}}$的解析解方法。

第7章侧重于介绍线性代数与矩阵理论在各个领域中的应用，包括在电工学、

力学与化学领域求解线性代数方程组的应用；在线性控制系统理论中系统的定性分析、系统模型转换与特殊线性矩阵微分方程解析解运算领域的应用；在数字图像处理领域线性代数在图像压缩、几何尺寸变换与旋转等方面的应用；在图论领域的应用，包括图的矩阵表示、最短路径问题与复杂控制系统模型化简等方面的应用；在差分方程求解与Markov链建模与计算领域的应用；在数据拟合与分析领域的应用，包括数据的线性回归拟合、数据的最小二乘拟合、主成分分析与数据降维领域的应用等。

读者如果认真学习了本书相关的理论基础与MATLAB使用技巧，很容易利用相关的工具在自己的研究中取得有益的研究成果。

本章习题

1.1 已知下面的联立方程，试写出其矩阵形式，得出并验证方程的解。

（1）$\begin{cases} x_1+2x_2+3x_3=0 \\ 4x_1+x_2-2x_3=0 \\ 3x_1+2x_2+x_3=2 \end{cases}$

（2）$\begin{cases} x_1+x_2+x_3+x_4+x_5=1 \\ 3x_1+2x_2+x_3+x_4-3x_5=2 \\ x_2+2x_3+2x_4+6x_5=3 \\ 5x_1+4x_2+3x_3+3x_4-x_5=4 \\ 4x_2+3x_3-5x_4=12 \end{cases}$

1.2 已知如下的矩阵型线性代数方程，试写出联立方程方程的形式。

（1）$\begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 2 \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 7 \end{bmatrix}$

（2）$\begin{bmatrix} 2 & 9 & 4 & 12 & 5 & 8 & 6 \\ 12 & 2 & 8 & 7 & 3 & 3 & 7 \\ 3 & 0 & 3 & 5 & 7 & 5 & 10 \\ 3 & 11 & 6 & 6 & 9 & 9 & 1 \\ 11 & 2 & 1 & 4 & 6 & 8 & 7 \\ 5 & -18 & 1 & -9 & 11 & -1 & 18 \\ 26 & -27 & -1 & 0 & -15 & -13 & 18 \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 1 & 9 \\ 5 & 12 \\ 4 & 12 \\ 10 & 9 \\ 0 & 5 \\ 10 & 18 \\ -20 & 2 \end{bmatrix}$

第2章

矩阵的表示与基本运算

如果需要求解线性代数问题，首先应该将矩阵输入计算机，只有将矩阵输入了计算机，才能对其进行运算，实现线性代数要求的各种问题的求解。2.1 节给出矩阵的基本概念，并介绍将一般矩阵输入 MATLAB 工作空间的方法。2.2 节介绍单位矩阵、随机矩阵、对角矩阵、Hankel 矩阵等特殊矩阵的输入方法。2.3 节介绍任意符号矩阵的输入方法，并介绍如何在 MATLAB 环境中直接建立常数型任意矩阵与函数型任意矩阵。2.4 节介绍稀疏矩阵的输入方法、存储方式与转换方法，并介绍稀疏矩阵的非零元素的提取与图形表示方法。2.5 节介绍矩阵的基本运算方法，包括简单的代数运算与复数矩阵的处理。2.6 节介绍矩阵的微积分运算。

2.1 一般矩阵的输入方法

矩阵是线性代数中最基本的数学单元。本节将给出矩阵的一般数学形式，然后介绍实数矩阵与复数矩阵的输入方法。

定义 2-1 一个 n 行 m 列的矩阵又称为 $n \times m$ 矩阵，其数学形式为

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \tag{2-1-1}$$

定义 2-2 如果矩阵 $\boldsymbol{A}$ 为 $n \times m$ 实矩阵，则记作 $\boldsymbol{A} \in \mathscr{R}^{n \times m}$，复矩阵则记作 $\boldsymbol{A} \in \mathscr{C}^{n \times m}$。

定义 2-3 一个 $n \times 1$ 的矩阵称为列向量，一个 $1 \times n$ 的矩阵称为行向量，1×1 的矩阵称为标量。

在 MATLAB 中，矩阵是最基本的数据单元，可以用简单的命令输入一个矩阵，也可以用简单的命令进行矩阵分析与线性代数计算。矩阵可以由双精度型数据结构表示，也可以由符号型数据结构表示。这两种方法在描述与求解矩阵问题上是不

同的，前者往往对应于矩阵问题的数值解，后者对应于矩阵问题的解析解。

例2-1　试将下面的实数矩阵输入MATLAB环境。

$$\boldsymbol{A}=\begin{bmatrix}-1 & 5 & 4 & 6\\ 0 & 2 & 4 & -2\\ 4 & 0 & -2 & 5\end{bmatrix}$$

解　可以将整个矩阵逐行输入MATLAB环境，同一行的元素由逗号或空格分隔，逗号或空格的作用是完全一致的；换行用真正的换行键表示，也可以由分号分隔，其作用是一致的。上述$\boldsymbol{A}$矩阵可以由下面的语句直接输入MATLAB环境。

```
>> A=[-1,5,4,6; 0,2,4,-2; 4,0,-2,5], B=sym(A)
```

正常情况下，由以上命令输入的矩阵默认是用双精度数据结构进行存储的，如果想将其转换成符号型数据结构，则可以采用`sym()`命令进行转换。比如，上面得出的$\boldsymbol{A}$矩阵就是符号型的矩阵。请注意两种不同矩阵在显示格式上的区别。

例2-2　试将下面的复数矩阵输入MATLAB工作空间。

$$\boldsymbol{A}=\begin{bmatrix}-1+6\mathrm{j} & 5+3\mathrm{j} & 4+2\mathrm{j} & 6-2\mathrm{j}\\ \mathrm{j} & 2-\mathrm{j} & 4 & -2-2\mathrm{j}\\ 4 & -\mathrm{j} & -2+2\mathrm{j} & 5-2\mathrm{j}\end{bmatrix}$$

解　要输入复数矩阵，则应该学会使用`i`或`j`。例如，数学符号6j可以直接表示为`6i`或`6j`，这样就可以由下面的语句直接输入这个复数矩阵，得到的$\boldsymbol{A}$矩阵存储为双精度数值矩阵，$\boldsymbol{B}$矩阵为符号型矩阵。如果显示两个矩阵，则可见两种矩阵的显示格式也是不同的，读者可以尝试验证一下，体验不同的表示形式。

```
>> A=[-1+6i,5+3i,4+2i,6-2i; 1i,2-1i,4,-2-2i;
      4,-1i,-2+2i,5-2i]
   B=sym(A)
```

例2-3　输入复数矩阵时输入`j`时，应该使用`1i`而不要使用`i`表示，否则可能出现意想不到的结果。另外，下面的语句由于多余空格的加入会产生错误。

```
>> A=[-1 +6i,5+3i,4+2i,6-2i; 1i,2-1i,4,-2-2i;
      4,-1i,-2+2i,5-2i]
```

以上语句由于使用了多余的空格，MATLAB执行机制会将-1理解成第一个元素，`+6i`被理解成第二个元素，所以第一行就出现了5个元素，其他行有4个元素，故而最终导致错误信息。

2.2　特殊矩阵的输入方法

2.1节介绍了一般矩阵的输入方法，需要逐个元素将矩阵输入计算机。然而，对于一些特定的矩阵，有时没有必要逐个元素输入，可借助于MATLAB语言提供的特殊矩阵输入函数。本节将介绍特殊矩阵的输入方法。

2.2.1　零矩阵、幺矩阵及单位矩阵

定义2-4　在一般的矩阵理论中，把所有元素都为零的矩阵称为零矩阵；把元素全为1的矩阵称为幺矩阵；把主对角线元素均为1，而其他元素全部为0的方阵称为单位矩阵。这里进一步扩展单位矩阵的定义，使其为$m \times n$矩阵。

零矩阵、幺矩阵和扩展单位矩阵的MATLAB生成函数分别为

$\boldsymbol{A}$=zeros(n)，$\boldsymbol{B}$=ones(n)，$\boldsymbol{C}$=eye(n)　　%生成$n \times n$方阵

$\boldsymbol{A}$=zeros(m,n)；$\boldsymbol{B}$=ones(m,n)；$\boldsymbol{C}$=eye(m,n)　%生成$m \times n$矩阵

$\boldsymbol{A}$=zeros(size($\boldsymbol{B}$))　%生成和矩阵$\boldsymbol{B}$同样维数的矩阵

例2-4　试生成一个3×8的零矩阵，并生成一个同维的单位矩阵。

解　以下语句可以生成一个3×8的零矩阵$\boldsymbol{A}$，并可以生成一个和$\boldsymbol{A}$维数相同的扩展单位阵$\boldsymbol{B}$。可见，特殊矩阵的输入还是很容易的。

```
>> A=zeros(3,8)    %产生零矩阵
   B=eye(size(A)) %生成同样维数的扩展单位阵。
```

可以将下面两个矩阵输入MATLAB工作空间。

$$\boldsymbol{A}=\begin{bmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\end{bmatrix}$$

例2-5　函数zeros()和ones()还可用于多维数组的生成。例如，zeros(3,4,5)将生成一个3×4×5的三维数组，其元素全部为零。

2.2.2　随机元素矩阵

顾名思义，随机元素矩阵的各个元素是随机产生的。生成随机数有两类方法，一类方法是利用电子装置生成随机数，称为物理生成的随机数；另一类是用数学方法生成具有随机数性质的数据，这类方法生成的随机数称为伪随机数。和物理方法生成的随机数相比，伪随机数是可以重复的。

1) 均匀分布伪随机数

如果矩阵的随机元素满足$[0,1]$区间上的均匀分布，则可以由MATLAB函数rand()生成，其调用格式为

$\boldsymbol{A}$=rand(n)，　%生成$n \times n$阶标准均匀分布伪随机数方阵

$\boldsymbol{A}$=rand(n,m)，%生成$n \times m$阶标准均匀分布伪随机数矩阵

函数rand()还可以用于多维数组的生成。例如，$\boldsymbol{A}$=rand(5,4,6)生成一个三维随机数组。

更一般地，如果想生成(a,b)区间上均匀分布的随机数，则可以先生成$(0,1)$上均匀分布的随机数矩阵$\boldsymbol{V}$，再通过简单变换生成满足需要的矩阵$\boldsymbol{V}_1$。

$\boldsymbol{V}$=rand(n,m)，　$\boldsymbol{V}_1$=$a+(b-a)$*$\boldsymbol{V}$

例2-6　试生成一组50000个$[-2,3]$区间均匀分布的伪随机数，求其均值并绘制该区间内随机数据分布的直方图。

解　生成这样的伪随机数向量并不是难事，通过下面的命令就可以得出这组数的均值为$\bar{v}=0.495245337225170$，与理论值0.5比较接近。此外，还可以绘制出这组数据的直方图，如图2-1所示。可以看出，在各个子区间内数据的分布还是比较均匀的。

```
>> N=50000; a=-2; b=3; v=a+(b-a)*rand(1,N);
   m=mean(v)
   c=linspace(-2,3,10); histogram(v,c)
```

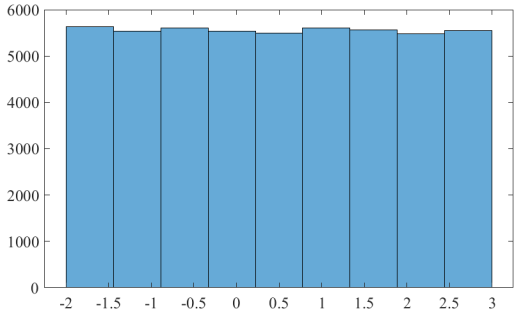


图 2-1　伪随机数的分布直方图

前面提到，伪随机数有两个特点，其一是由数学公式生成的，其二是可以重复的。如何获得可重复的伪随机数呢？生成伪随机数是需要种子（seed）的，如果种子相同，则生成的伪随机数也是相同的。MATLAB提供了控制量rng()函数描述或设置伪随机数种子，该函数的简单调用格式为s=rng，该命令读取当前使用的随机数生成信息。将s存储起来，下次想生成重复的随机数，则可以将s调用出来，再给出命令rng(s)，则可以获得重复的伪随机数。

例2-7　试生成两组完全一致的均匀分布伪随机数，并比较效果。

解　通常情况下调用两次rand()函数，是可以得出两组完全不同的伪随机数的，而执行下面的语句，生成随机数并将其存入数据文件datac27.mat。

```
>> s=rng; a=rand(1,100); save datac27 s
```

下次不管什么时候读入该文件都可以获得随机数种子s，这样，两次调用rand()函数，则得出的误差err为零，说明可以生成完全一致的伪随机数。

```
>> load datac27; rng(s); b=rand(1,100); err=a-b
```

2) 均匀分布的整数矩阵

利用MATLAB的内核函数randi()也可以生成在$[a,b]$区间上均匀分布的随

机整数矩阵，其调用格式为

$\boldsymbol{A}$=randi([a,b],[n,m]), $\boldsymbol{B}$=randi([a,b],n)

其中，a、b均应该为整数，且$a \leqslant b$，用于表示区间的下限与上限。前面给出的命令可以生成一个$n \times m$长方形矩阵$\boldsymbol{A}$，或$n \times n$方阵$\boldsymbol{B}$。

例2-8　试生成一个由0和1构成的10×10非奇异整数矩阵。

解　可以考虑用无限循环结构来生成这样的矩阵，若已经找到非奇异方阵，则用break命令终止循环。这里使用rank()函数用于求矩阵的秩，如果$\boldsymbol{A}$矩阵的秩为10(满秩)，则生成的10×10矩阵是非奇异矩阵。

```
>> while(1)
      A=randi([0,1],10); if rank(A)==10, break; end,
   end, A %找到并显示非奇异矩阵
```

调用上面的语句每次可能得到不同的矩阵。例如，可能得出下面的满秩矩阵。

$$\boldsymbol{A}=\begin{bmatrix}1&0&1&1&0&0&0&1&0&1\\1&1&0&0&0&1&1&0&1&1\\1&0&0&1&1&0&1&1&0&0\\1&1&0&1&0&1&0&1&0&0\\0&1&0&0&0&1&0&0&0&0\\0&0&0&1&1&0&0&0&1&0\\0&0&0&0&1&1&1&0&0&0\\1&0&0&0&1&1&1&1&1&1\\0&0&0&0&0&1&0&0&1&1\\1&1&0&0&0&1&0&0&0&0\end{bmatrix}$$

3) 正态分布伪随机数

满足标准正态分布$N(0,1)$的随机数矩阵可以直接由函数randn()生成，该函数的调用格式与rand()函数一致。当然，也可以使用命令$\boldsymbol{B}$=randn(size($\boldsymbol{A}$))的形式调用该函数，生成一个与$\boldsymbol{B}$矩阵同维数的标准正态分布伪随机数矩阵。

若想生成满足$\mathrm{N}(\mu,\sigma^2)$的正态分布的随机数，则可以先用$\boldsymbol{V}$=randn(n,m)命令生成标准正态分布的随机数矩阵$\boldsymbol{V}$，再用$\boldsymbol{V}_1=\mu+\sigma*\boldsymbol{V}$命令就可以转换成所需的矩阵。

其实，除了这两类分布的随机数外，还可以调用random()函数生成其他分布的随机数，读者可以自己尝试该函数的调用方法。

2.2.3　Hankel矩阵

Hankel矩阵是以德国数学家Hermann Hankel（1839−1873）命名的一类特殊矩阵，其具体形式与生成方法如下。

定义2-5　Hankel矩阵的数学表达式为

$$\boldsymbol{H}=\begin{bmatrix} c_1 & c_2 & \cdots & c_m \\ c_2 & c_3 & \cdots & c_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n+1} & \cdots & c_{n+m-1} \end{bmatrix} \tag{2-2-1}$$

如果$n\to\infty$，则可以构造无穷维Hankel矩阵，不过MATLAB只能处理有限维矩阵。Hankel矩阵是对称矩阵，特点是每条反对角线上所有的元素都相同。

在MATLAB语言中提供了Hankel矩阵生成函数`hankel()`，该函数可以有两种调用方法：

$\boldsymbol{H}_1$=`hankel(`$\boldsymbol{c}$`)`，　$\boldsymbol{H}_2$=`hankel(`$\boldsymbol{c}$`,`$\boldsymbol{r}$`)`

其中，给定两个向量$\boldsymbol{c}$和$\boldsymbol{r}$，分别生成两种不同的Hankel矩阵。

$$\boldsymbol{H}_1=\begin{bmatrix} c_1 & c_2 & \cdots & c_n \\ c_2 & c_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_n & 0 & \cdots & 0 \end{bmatrix},\quad \boldsymbol{H}_2=\begin{bmatrix} c_1 & c_2 & \cdots & \cdot \\ c_2 & c_3 & \cdots & \cdot \\ \vdots & \vdots & \ddots & \vdots \\ c_n & r_2 & \cdots & r_m \end{bmatrix}$$

第一种调用格式比较简单，生成的是一个方阵，矩阵下三角的元素都是零，其余元素由“Hankel矩阵反对角线元素相同”这一性质可以逐项填写出来。在第二种调用格式下，要求$c_n=r_1$，否则将给出错误信息，自动略去r_1的值，故而得出的Hankel矩阵为$\boldsymbol{H}_2$的形式。但是，右上角元素是c还是r将完全取决于$\boldsymbol{c}$向量与$\boldsymbol{r}$向量哪个长。如果$n=m$，则生成方阵，这时右上角元素为c_n。

例2-9　试用MATLAB语句输入下面两个Hankel矩阵。

$$\boldsymbol{H}_1=\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix},\quad \boldsymbol{H}_2=\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 0 \\ 3 & 0 & 0 \end{bmatrix}$$

解　分析给出的矩阵，可以用向量分别表示该矩阵的首列和最后一行，$\boldsymbol{c}$=`[1,2,3]`，$\boldsymbol{r}$=`[3,4,5,6,7,8,9]`，则可以由下面语句生成所需的Hankel矩阵。注意，两个向量中3这个共同元素。

```
>> c=[1 2 3]; r=[3 4 5 6 7 8 9];
   H1=hankel(c,r), H2=hankel(c) %Hankel矩阵输入
```

2.2.4　对角元素矩阵

对角矩阵是一种特殊的矩阵，这种矩阵的主对角线元素可以为零或非零元素，而非对角线元素的值均为零。对角矩阵的数学描述方法为$\mathrm{diag}(\alpha_1,\alpha_2,\cdots,\alpha_n)$。其

中，对角矩阵的矩阵表示为

$$\mathrm{diag}(\alpha_1,\alpha_2,\cdots,\alpha_n)=\begin{bmatrix}\alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_n\end{bmatrix} \tag{2-2-2}$$

MATLAB 提供了对角矩阵的生成函数 `diag()`。该函数的调用格式为

$\boldsymbol{A}$=`diag(`$\boldsymbol{v}$`)` %已知向量，生成对角矩阵

$\boldsymbol{v}$=`diag(`$\boldsymbol{A}$`)` %已知矩阵，提取对角元素列向量

$\boldsymbol{A}$=`diag(`$\boldsymbol{v}$`,`k`)` %生成第 k 条对角线为 $\boldsymbol{v}$ 的矩阵，若 $\boldsymbol{v}$ 为矩阵则提取该对角线

MATLAB 提供的 `diag()` 函数是一个很有特色的函数，它不但能输入对角矩阵，也可以指定某条对角线，输入次对角矩阵，还可以从一个矩阵提取对角或次对角元素。下面通过例子演示该函数的使用方法。

例 2-10　试将下面三个矩阵输入 MATLAB 工作空间。

$$\boldsymbol{A}=\begin{bmatrix}1&0&0\\0&2&0\\0&0&3\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0&0&1&0&0\\0&0&0&2&0\\0&0&0&0&3\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}0&0&0&0\\1&0&0&0\\0&2&0&0\\0&0&3&0\end{bmatrix}$$

解　可以先将 $\boldsymbol{v}=[1,2,3]$ 向量输入 MATLAB 工作空间，这样，矩阵 $\boldsymbol{A}$ 可以直接用 `diag()` 函数生成。现在考虑矩阵 $\boldsymbol{B}$，由于该矩阵主对角线上第 2 条对角线为 $\boldsymbol{v}$ 向量，所以应该将 k 设置成 2，而矩阵 $\boldsymbol{C}$ 是主对角线下一条对角线的元素为 $\boldsymbol{v}$，所以可以将 k 设置为 -1。这样，由下面的语句可以将这三个矩阵直接输入 MATLAB 工作空间。

```
>> v=[1 2 3]; A=diag(v), B=diag(v,2), C=diag(v,-1)
```

如果采用下面的命令还可以从矩阵提取出对角元素，但是得出的是列向量。

```
>> v1=diag(A), v2=diag(B,2), v3=diag(C,-1)
```

例 2-11　n 阶 Jordan 矩阵的一般形式如下，试编写 MATLAB 函数构造该矩阵。

$$\boldsymbol{J}=\begin{bmatrix}-\alpha&1&0&\cdots&0\\0&-\alpha&1&\cdots&0\\0&0&-\alpha&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&-\alpha\end{bmatrix}$$

解　可以将 Jordan 矩阵分解为两个矩阵的和，一个是 $\alpha\boldsymbol{I}$，其中 $\boldsymbol{I}$ 为单位矩阵；另一个为第一次对角线元素均为 1 的矩阵，又称为幂零矩阵，后面将详细介绍。有了这样的想法，不难由下面的函数结构直接生成 $n\times n$ 的 Jordan 矩阵。

```
function J=jordan_matrix(alpha,n)
v=ones(1,n-1); J=alpha*eye(n)+diag(v,1);
```

定义2-6　若$\boldsymbol{A}_1,\boldsymbol{A}_2,\cdots,\boldsymbol{A}_n$为已知矩阵，则块对角矩阵的数学定义为

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{A}_1 & & & \\ & \boldsymbol{A}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{A}_n\end{bmatrix} \tag{2-2-3}$$

可以编写一个diagm()函数，由给出的子矩阵构造出块对角矩阵。该函数的调用格式为$\boldsymbol{A}$=diagm($\boldsymbol{A}_1,\boldsymbol{A}_2,\cdots,\boldsymbol{A}_n$)，该函数允许输入任意多个子矩阵。

```
function A=diagm(varargin), A=[];
for i=1:length(varargin), A1=varargin{i}; %用循环结构构造块对角矩阵
   [n,m]=size(A); [n1,m1]=size(A1); A(n+1:n+n1,m+1:m+m1)=A1;
end
```

其实，MATLAB提供的blkdiag()函数也能实现同样的功能，并且其功能更强大。除了一般矩阵，还可以直接处理符号型矩阵与稀疏矩阵的块对角矩阵生成。

例2-12　假设已知下面的子矩阵，试构造出块对角矩阵。

$$\boldsymbol{A}=\begin{bmatrix}8&1&6\\3&5&7\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}1&3\\4&2\end{bmatrix}$$

解　可以使用两种方法构造块对角矩阵。

```
>> A=[8,1,6; 3,5,7]; B=[1,3; 4,2];
   C=blkdiag(A,B), D=diagm(A,B)
```

得出的结果是完全一致的。

$$\boldsymbol{C}=\boldsymbol{D}=\left[\begin{array}{ccc:cc}8&1&6&0&0\\3&5&7&0&0\\ \hdashline 0&0&0&1&3\\0&0&0&4&2\end{array}\right]$$

如果矩阵$\boldsymbol{B}$是符号型矩阵，则下面得出的矩阵$\boldsymbol{C}$为符号型矩阵，矩阵$\boldsymbol{D}$为双精度矩阵，其内容是相同的。

```
>> B=sym([1,3; 4,2]); C=blkdiag(A,B), D=diagm(A,B)
```

2.2.5　Hilbert矩阵及Hilbert逆矩阵

Hilbert矩阵是以德国数学家David Hilbert（1862−1943）命名的特殊矩阵。

定义2-7　Hilbert矩阵是一类特殊矩阵，它的第(i,j)个元素的值满足$h_{i,j}=1/(i+j-1)$。一个n阶的Hilbert矩阵可以写成

$$\boldsymbol{H}=\begin{bmatrix}1&1/2&1/3&\cdots&1/n\\1/2&1/3&1/4&\cdots&1/(n+1)\\\vdots&\vdots&\vdots&\ddots&\vdots\\1/n&1/(n+1)&1/(n+2)&\cdots&1/(2n-1)\end{bmatrix} \tag{2-2-4}$$

产生Hilbert矩阵的MATLAB函数为$\boldsymbol{A}$=hilb(n)。其中，n为要产生的矩阵阶次，生成的矩阵是$n \times n$方阵。

高阶Hilbert矩阵一般为坏条件的矩阵，直接对其求逆往往会产生浮点溢出的现象。MATLAB提供了直接求取Hilbert逆矩阵的函数$\boldsymbol{B}$=invhilb(n)。由于Hilbert矩阵本身接近奇异的性质，所以在处理该矩阵时建议尽量采用符号运算工具箱，而采用数值解时应该检验结果的正确性。

例2-13　如果想输入一个长方形的Hilbert矩阵，最简单、最高效的方法是生成一个Hilbert方阵，然后从中提取所需的长方形矩阵。当然，也可以通过下面的命令生成长方形Hilbert矩阵。

```
>> m=10000; n=5;
   [x,y]=meshgrid(1:m,1:n); H=1./(x+y-1);
```

2.2.6　相伴矩阵

定义2-8　假设有一个首一化（monic）的多项式

$$p(s) = s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_{n-1} s + a_n \tag{2-2-5}$$

则可以写出一个相伴（companion）矩阵（或称友矩阵）

$$\boldsymbol{A}_{\mathrm{c}} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{2-2-6}$$

生成相伴矩阵的MATLAB函数调用格式为$\boldsymbol{A}_{\mathrm{c}}$=compan($\boldsymbol{a}$)。其中，$\boldsymbol{a}$为降幂排列的多项式系数向量，该函数将自动对多项式进行首一化处理。

例2-14　考虑一个多项式$P(s) = 2s^4 + 4s^2 + 5s + 6$，试写出该多项式的相伴矩阵。

解　先输入特征多项式，则相伴矩阵可以通过下面的语句建立起来，赋给$\boldsymbol{A}$矩阵。

```
>> P=[2 0 4 5 6]; A=compan(P) %给出向量，自动首一化，可以建立相伴矩阵
```

以上语句可以得出相伴矩阵为

$$\boldsymbol{A} = \begin{bmatrix} 0 & -2 & -2.5 & -3 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2.2.7　Wilkinson矩阵

Wilkinson矩阵是以英国数学家James Hardy Wilkinson（1919−1986）命名的测试矩阵。

定义 2-9　Wilkinson矩阵是三对角矩阵，其第1与第-1条对角线元素都是1，$2m+1$阶Wilkinson矩阵主对角线为

$$\boldsymbol{v}=\left[m,m-1,\cdots,2,1,0,1,2,\cdots,m-1,m\right] \tag{2-2-7}$$

$2m$阶的主对角元素为

$$\boldsymbol{v}=\left[(2m-1)/2,\cdots,3/2,1/2,1/2,3/2,\cdots,(2m-1)/2\right] \tag{2-2-8}$$

Wilkinson矩阵可以由$\boldsymbol{W}$=`wilkinson(`n`)`函数直接生成，若需要符号型Wilkinson矩阵，则需要调用`sym()`命令进行转换。

例 2-15　试生成5阶和6阶Wilkinson矩阵。

解　可以用下面的语句直接生成所需的Wilkinson矩阵。其中，第一个Wilkinson矩阵为双精度矩阵，第二个Wilkinson矩阵为转换后的符号型矩阵。

```
>> W1=wilkinson(5), W2=sym(wilkinson(6))
```

得出的结果为

$$\boldsymbol{W}_1=\begin{bmatrix}2&1&0&0&0\\1&1&1&0&0\\0&1&0&1&0\\0&0&1&1&1\\0&0&0&1&2\end{bmatrix},\ \boldsymbol{W}_2=\begin{bmatrix}5/2&1&0&0&0&0\\1&3/2&1&0&0&0\\0&1&1/2&1&0&0\\0&0&1&1/2&1&0\\0&0&0&1&3/2&1\\0&0&0&0&1&5/2\end{bmatrix}$$

2.2.8　Vandermonde矩阵

Vandermonde矩阵是以法国数学家Alexandre-Théophile Vandermonde（1735$-$1796）命名的一类特殊矩阵。

定义 2-10　给定向量$\boldsymbol{c}=[c_1,c_2,\cdots,c_n]$，可以写出一个矩阵，其第$(i,j)$个元素满足$v_{i,j}=c_i^{n-j}(i,j=1,2,\cdots,n)$。这样构成的矩阵称为Vandermonde矩阵，其数学形式为

$$\boldsymbol{V}=\begin{bmatrix}c_1^{n-1}&c_1^{n-2}&\cdots&c_1&1\\c_2^{n-1}&c_2^{n-2}&\cdots&c_2&1\\\vdots&\vdots&\ddots&\vdots&\vdots\\c_n^{n-1}&c_n^{n-2}&\cdots&c_n&1\end{bmatrix} \tag{2-2-9}$$

可以由MATLAB提供的$\boldsymbol{V}$=`vander(`$\boldsymbol{c}$`)`函数生成一个Vandermonde矩阵。

例 2-16　试建立如下的Vandermonde矩阵。

$$\boldsymbol{V}=\begin{bmatrix}1&1&1&1&1\\1&2&3&4&5\\1&4&9&16&25\\1&8&27&64&125\\1&16&81&256&625\end{bmatrix}$$

解　这里要求的矩阵与式(2-2-9)给出的形式不完全一致，是Vandermonde标准型的转置。可以首先生成向量$\boldsymbol{c}$=[1,2,3,4,5]，得出其Vandermonde标准型后再将其逆时针90°旋转，则可以得出所需$\boldsymbol{V}$矩阵。

```
>> c=[1, 2, 3, 4, 5]; V=vander(c); V=rot90(V) %先建立标准矩阵再旋转
```

2.2.9　一些常用的测试矩阵

定义 2-11　如果一个$n \times n$矩阵的每行元素、每列元素、正反对角线元素的和都是一个常数，则该矩阵称为魔方矩阵(magic matrix)。

定义 2-12　如果一个矩阵的第一行、第一列的元素都是1，且其余元素可以由$a_{ij} = a_{i,j-1} + a_{i-1,j}$格式递推地生成，则该矩阵称为Pascal矩阵。

定义 2-13　正向对角线上元素都相同的矩阵称为Toeplitz矩阵，所以已知矩阵的第一行$\boldsymbol{r}$与第一列$\boldsymbol{c}$，即可以构造出Toeplitz矩阵。

$$\boldsymbol{T} = \begin{bmatrix} c_1 & r_2 & r_3 & \cdots & r_m \\ c_2 & c_1 & r_2 & \cdots & r_{m-1} \\ c_3 & c_2 & c_1 & \cdots & r_{m-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n-1} & c_{n-2} & \cdots & \cdot \end{bmatrix} \tag{2-2-10}$$

Pascal矩阵是以法国数学家Blaise Pascal(1623−1662)命名的特殊矩阵，Toeplitz矩阵是以德国数学家Otto Toeplitz(1881−1940)命名的特殊矩阵。

MATLAB提供的一些函数可以生成特殊矩阵。

$\boldsymbol{M}$=magic(n)，$\boldsymbol{P}$=pascal(n)，$\boldsymbol{T}$=toeplitz($\boldsymbol{r}$,$\boldsymbol{c}$)

例 2-17　试分别生成四阶魔方矩阵、Pascal矩阵和Toeplitz矩阵。

解　由下面的命令可以直接生成所需的矩阵。

```
>> A=magic(4), B=pascal(4), C=toeplitz(1:4)
```

可以直接得到下面的结果。

$$\boldsymbol{A} = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

观察Pascal矩阵的上三角部分，如果从第一列的每个元素沿反向对角线方向看，则相应的各个元素都是二次项系数。

MATLAB还提供了gallery()函数生成一些特殊矩阵，其调用格式为

$\boldsymbol{A}$=gallery(矩阵类型,n,其他参数)

其中，“矩阵类型”可以为'binomial'、'cauchy'和'chebspec'等选项，具体可以用doc gallery命令查询。

2.3　符号型矩阵的输入方法

前面介绍过，若已建立起数值矩阵$\boldsymbol{A}$，则可以由$\boldsymbol{B}$=sym($\boldsymbol{A}$)语句将其转换成符号型矩阵。所有数值矩阵均可以通过这种形式转换成符号型矩阵，可以利用符号运算工具箱获得更高精度的解。相反地，一个全数值的符号型矩阵$\boldsymbol{B}$可以通过命令$\boldsymbol{A}_1$=double($\boldsymbol{B}$)转换成双精度矩阵$\boldsymbol{A}_1$。

但是，如果矩阵中含有符号变量，则不能由double()函数进行转换，否则将给出错误信息。

2.3.1　特殊符号矩阵的输入方法

前面介绍的很多特殊矩阵输入函数都可以直接生成符号型矩阵，或由sym()函数转换成符号型矩阵。例如，eye(5)函数可以生成双精度单位矩阵，然后由sym()函数转换成符号型单位矩阵。较新版本的MATLAB还支持一些特殊符号矩阵，例如Vandermonde矩阵、Hankel矩阵和相伴矩阵等的直接输入函数。如果读者使用的版本不支持生成这样的符号型矩阵，则可以使用本书所附的函数vandersym()、hankelsym()和compansym()生成相应的符号型矩阵。

例2-18　试由多项式$P(\lambda)=a_1\lambda^7+a_2\lambda^6+a_3\lambda^5+\cdots+a_6\lambda^2+a_7\lambda+a_8$建立相伴矩阵。

解　可以先由sym()函数生成行向量$\boldsymbol{a}$，然后由compan()函数直接生成所需的相伴矩阵。这里，由于原始的向量不是首一化的系数向量，MATLAB函数会自动作首一化处理，然后再生成相伴矩阵。

```
>> syms a1 a2 a3 a4 a5 a6 a7 a8;
   a=[a1 a2 a3 a4 a5 a6 a7 a8]; A=compan(a) %建立如下符号型相伴矩阵
```

得出的相伴矩阵为

$$\boldsymbol{A}=\begin{bmatrix}-a_2/a_1 & -a_3/a_1 & -a_4/a_1 & -a_5/a_1 & -a_6/a_1 & -a_7/a_1 & -a_8/a_1\\ 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$$

2.3.2　任意常数矩阵的输入

MATLAB提供的sym()函数除了矩阵类型转换外，还可以生成任意元素a_{ij}构成的矩阵，具体的调用格式为

$\boldsymbol{A}$=sym('a%d%d',[n,m])　%其中，%d%d表示双下标

上面的语句可以生成$n\times m$的任意矩阵，其元素为a_{ij}，$i=1,2,\cdots,n$，$j=$

$1, 2, \cdots, m$。如果不给出m，则将生成一个$n \times n$的任意方阵。

类似地，使用下面的命令可以生成任意的行向量$\boldsymbol{v}_1$和列向量$\boldsymbol{v}_2$。

$\boldsymbol{v}_1$=sym('a',[1,n]), $\boldsymbol{v}_2$=sym('a',[n,1]), %第一种方法

$\boldsymbol{v}_1$=sym('a%d',[1,n]), $\boldsymbol{v}_2$=sym('a%d',[n,1]), %第二种方法

例2-19　试输入如下的两个矩阵和一个列向量。

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \\ f_{31} & f_{32} \\ f_{41} & f_{42} \end{bmatrix},\quad \boldsymbol{v}=\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}$$

解　从给出的要求可见，矩阵$\boldsymbol{A}$和$\boldsymbol{B}$都需要双下标，所以需要给出字符串'%d%d'设置。另外，两个矩阵的符号分别为a和f，所以可以用下面的语句直接输入两个矩阵。相比之下，向量$\boldsymbol{v}$的输入比较简单与直观。

```
>> A=sym('a%d%d',4), B=sym('f%d%d',[4,2]), v=sym('v',[4,1])
```

如果想进一步声明满足某种属性的矩阵，还可以用assume()与assumeAlso()函数设定。例如，可以使用下面的命令设置矩阵属性。

```
>> assume(A,'real'); assumeAlso(A,'integer') %设置其他矩阵属性
```

其中，这两个函数可以使用的属性为integer（整数）、rational（有理数）、real（实数）与positive（正数）。如果不特别设置，则一般矩阵的默认类型为复数矩阵。

例2-20　试重新输入例2-18中要求的相伴矩阵。

解　由$\boldsymbol{a}$=sym('a',[1,8])命令就可以声明任意符号型向量，而无须像例2-18那样逐个声明、逐个输入向量元素，即使更大规模的向量也可以这样输入。

```
>> a=sym('a',[1,20]); A=compan(a) %建立符号型相伴矩阵
```

例2-21　试建立一个3×6的任意实有理数矩阵。

解　由前面的介绍，不难建立所需的矩阵。

```
>> A=sym('a%d%d',[3,6]);
   assume(A,'real'), assumeAlso(A,'rational')
```

2.3.3　任意矩阵函数的输入

sym()函数可以生成任意常数矩阵，根据该函数还可以编写出任意矩阵函数的输入函数。例如，如果想生成矩阵函数$\boldsymbol{M}=\{m_{ij}(x,y)\}$，则可以根据需要编写出通用函数any_matrix()。

```
function A=any_matrix(nn,sA,varargin) %生成任意矩阵
v=varargin; n=nn(1); if length(nn)==1, m=n; else, m=nn(2); end
s=''; k=length(v); K=0; if n==1 || m==1, K=1; end
if k>0, s='('; for i=1:k, s=[s ',' char(v{i})]; end
s(2)=[]; s=[s ')']; end
```

```
for i=1:n, for j=1:m %用循环结构逐个元素单独处理
   if K==0, str=[sA int2str(i),int2str(j)];
   else, str=[sA int2str(i*j)]; end
   eval(['syms ' str s]); eval(['A(i,j)=' str ';']); %指定矩阵元素
end, end
```

该函数的调用格式为

$\boldsymbol{A}(x_1,x_2,\cdots,x_k)$=any_matrix([$n$,$m$],'a',$x_1,x_2,\cdots,x_k$)

其中，n，m为矩阵的行数与列数，如果只给出n，将生成一个方阵。变元x_1，x_2，$\cdots$，x_k为事先声明的符号变量，a还可以使用任何其他字母，生成的函数矩阵为

$$\boldsymbol{A}=\begin{bmatrix} a_{11}(x_1,x_2,\cdots,x_k) & \cdots & a_{1m}(x_1,x_2,\cdots,x_k) \\ \vdots & \ddots & \vdots \\ a_{n1}(x_1,x_2,\cdots,x_k) & \cdots & a_{nm}(x_1,x_2,\cdots,x_k) \end{bmatrix}$$

例2-22　若先声明符号变量x、y和t，则可以用下面的命令生成函数矩阵。注意，使用矩阵函数的方式输入矩阵。

```
>> syms x y t; clear v X
   v(x,y)=any_matrix([3,1],'a',x,y), X(t)=any_matrix(3,'m',t)
```

生成的任意函数矩阵与向量为

$$\boldsymbol{v}(x,y)=\begin{bmatrix} a_1(x,y) \\ a_2(x,y) \\ a_3(x,y) \end{bmatrix},\ \boldsymbol{X}(t)=\begin{bmatrix} m_{11}(t) & m_{12}(t) & m_{13}(t) \\ m_{21}(t) & m_{22}(t) & m_{23}(t) \\ m_{31}(t) & m_{32}(t) & m_{33}(t) \end{bmatrix}$$

2.4　稀疏矩阵的输入

在很多应用中经常需要描述一些特殊的大型矩阵，而这类矩阵的大部分元素都是零，仅有少部分非零元素，这样的矩阵称为稀疏矩阵（sparse matrix）。若选择合适的求解算法，稀疏矩阵的计算比常规矩阵效率更高。MATLAB支持稀疏矩阵的输入，且很多矩阵分析函数支持稀疏矩阵的特别处理。

稀疏矩阵可以由sparse()函数读入MATLAB。$\boldsymbol{A}$=sparse($\boldsymbol{p}$,$\boldsymbol{q}$,$\boldsymbol{w}$)，其中，$\boldsymbol{p}$、$\boldsymbol{q}$为非零元素的行号和列号构成的向量，$\boldsymbol{w}$为相应位置的矩阵元素构成的向量。这三个向量的长度是一致的，否则将给出错误信息。

对双精度数据结构而言，存储一个矩阵元素需要8字节，所以要存储一个$n\times n$的方阵需要$8n^2$字节。稀疏矩阵要存储一个非零元素需要8字节存储该元素本身，还需要16字节存储其所在的行数与列数，所以总共需要$24m$字节存储非零元素，m为非零元素的个数。可见，$m=n^2/3$时，存储双精度矩阵与稀疏矩阵所需的空间是一致的。换句话说，如果一个矩阵2/3以上的元素为零，则利用稀疏矩阵的方式存储

矩阵比较经济，且矩阵的稀疏度越高存储越经济。

常规矩阵 $\boldsymbol{B}$ 与稀疏矩阵 $\boldsymbol{A}$ 是可以相互转换的，也可以检验一个矩阵 $\boldsymbol{A}$ 是不是按稀疏矩阵的形式存储的，这些转换与检测函数的调用格式为

$\boldsymbol{B}$=full($\boldsymbol{A}$)，　$\boldsymbol{A}$=sparse($\boldsymbol{B}$)，key=issparse($\boldsymbol{A}$)

一个常规矩阵或稀疏矩阵的非零元素个数可以由 n=nnz($\boldsymbol{A}$)直接得出。如果想提取出矩阵 $\boldsymbol{A}$ 中全部的非零元素，则可以使用 $\boldsymbol{v}$=nonzeros($\boldsymbol{A}$)命令，其提取顺序是按列提取。

稀疏矩阵 $\boldsymbol{A}$ 的非零元素所在的位置可以用spy($\boldsymbol{A}$)函数显示出来，如果 $\boldsymbol{A}$ 不是稀疏矩阵，仍然能用该函数显示矩阵的非零元素。在spy()调用语句中，$\boldsymbol{A}$ 还可以是符号型矩阵。

例2-23　考虑例1-5中给出的简单梁结构，试分析矩阵的稀疏度。

解　这里只考虑双精度矩阵表示，不再采用符号型矩阵。可以用下面的语句重新输入原始矩阵，则可以得出该矩阵非零元素的个数为30。可以将该矩阵转换为稀疏矩阵，并绘制出稀疏矩阵的非零元素位置图。如图2-2所示，图中非零元素是用圆点表示的，没有圆点的位置元素都是零。

```
>> alpha=1/sqrt(2);
   A=[0  1  0  0  0  -1  0  0  0  0  0  0  0
      0  0  1  0  0  0  0  0  0  0  0  0  0
      alpha  0  0  -1  -alpha  0  0  0  0  0  0  0  0
      alpha  0  1  0  alpha  0  0  0  0  0  0  0  0
      0  0  0  1  0  0  0  -1  0  0  0  0  0
      0  0  0  0  0  0  1  0  0  0  0  0  0
      0  0  0  0  alpha  1  0  0  -alpha  -1  0  0  0
      0  0  0  0  1  0  1  0  alpha  0  0  0  0
      0  0  0  0  0  0  0  0  0  1  0  0  -1
      0  0  0  0  0  0  0  0  0  0  1  0  0
      0  0  0  0  0  0  0  1  -alpha  0  0  -alpha  0
      0  0  0  0  0  0  0  0  alpha  0  1  alpha  0
      0  0  0  0  0  0  0  0  0  0  0  alpha  1];
   nnz(A), B=sparse(A), spy(B)
```

给出whos命令，还可以比较二者的存储空间使用情况。

```
>> whos A B  %显示结果如下
  Name        Size              Bytes  Class     Attributes
  A          13x13              1352  double
  B          13x13               592  double    sparse
```

从得出的结果看，该矩阵的稀疏度不是很高。不过可以看出，在复杂的梁结构中，例

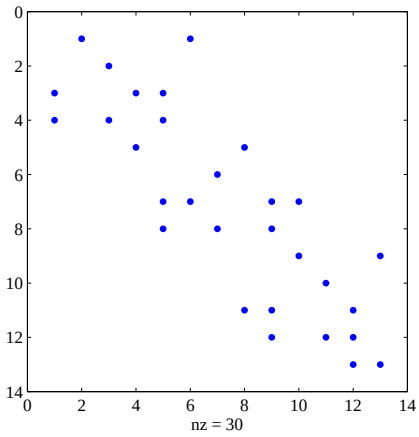


图 2-2　稀疏矩阵非零元素的分布

如有1000根梁，则建立起来的矩阵比较适合由稀疏矩阵表示，存储该矩阵会节省很多存储空间。

MATLAB还提供了一些特殊稀疏矩阵的输入方法。例如，可以由speye()函数直接输入单位矩阵；还可以由sprandn()与sprandsym()函数生成随机的稀疏矩阵。这两个矩阵的元素都满足标准正态分布，不同的是，后者生成的是对称矩阵。这些函数的调用格式为

E=speye([n,m])

A=sprandn(n,m,稀疏度)

B=sprandn(n,稀疏度)

例2-24　试生成一个稀疏度为0.0002%的50000×50000随机矩阵，并观察随机元素的分布。试将该矩阵转换为常规矩阵。

解　可以由下面的语句直接生成所需的稀疏矩阵，并绘制出5000个非零元素的示意图，如图2-3所示。若想将矩阵转换成常规矩阵，则得出“Out of memory”错误信息，因为MATLAB不能存储这样大规模的常规矩阵。使用稀疏矩阵的方式，即使稀疏度增大为1%，仍能存储该矩阵。

```
>> A=sprandn(50000,50000,0.0002/100); spy(A), nnz(A)
   whos A, B=full(A)
```

则得出的结果显示如下

```
  Name          Size              Bytes    Class    Attributes
   A          50000x50000         480008  double     sparse
```

如何求解两个稀疏矩阵的乘法至今是一个具有挑战性的问题。如果不能很好地求解乘法问题，则需要将稀疏矩阵转换成普通矩阵再进行乘法运算，不能利用稀疏矩阵自身的性质减少运算量。

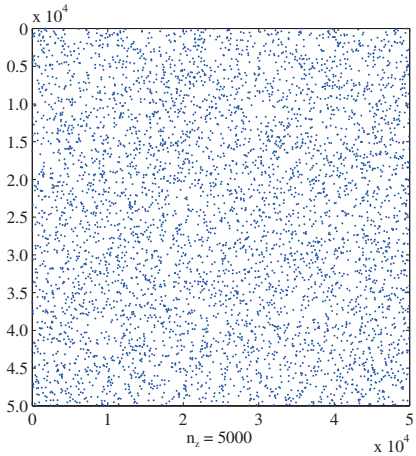


图 2-3　稀疏矩阵非零元素的分布

2.5　矩阵的基本运算

矩阵的基本运算是线性代数的基础。本节先介绍一些复数矩阵的变换方法，然后介绍一般矩阵的转置与翻转处理、矩阵的 Kronecker 和与乘积，最后介绍矩阵的代数运算方法。

2.5.1　复数矩阵的处理

MATLAB 可以直接表示复数矩阵。假设已知一个复数矩阵 $\boldsymbol{Z}$，则可以使用简单函数对该矩阵进行如下变换：

（1）**共轭复数矩阵**，$\boldsymbol{Z}_1$=conj($\boldsymbol{Z}$)；

（2）**实部、虚部提取**，$\boldsymbol{R}$=real($\boldsymbol{Z}$)，$\boldsymbol{I}$=imag($\boldsymbol{Z}$)；

（3）**幅值、相位表示**，$\boldsymbol{A}$=abs($\boldsymbol{Z}$)，$\boldsymbol{P}$=angle($\boldsymbol{Z}$)，其中相位的单位为弧度（radian，简记 rad）。

其实，这里的 $\boldsymbol{Z}$ 并不局限于矩阵，还可以是多维数组或符号表达式。

例 2-25　考虑下面给出的复数矩阵 $\boldsymbol{A}$，试提取矩阵的实部、虚部与共轭矩阵。

$$\boldsymbol{A}=\begin{bmatrix}1+9\mathrm{j} & 2+8\mathrm{j} & 3+7\mathrm{j}\\ 4+6\mathrm{j} & 5+5\mathrm{j} & 6+4\mathrm{j}\\ 7+3\mathrm{j} & 8+2\mathrm{j} & \mathrm{j}\end{bmatrix}$$

解　可以先输入复数矩阵，然后提取其实部与虚部矩阵。

```
>> A=[1+9i,2+8i,3+7j; 4+6j 5+5i,6+4i; 7+3i,8+2j 1i];
   R=real(A), I=imag(A), C=conj(A)
```

实部、虚部与共轭复数矩阵分别为

$$\boldsymbol{R}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix},\ \boldsymbol{I}=\begin{bmatrix}9&8&7\\6&5&4\\3&2&1\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}1-9\mathrm{j}&2-8\mathrm{j}&3-7\mathrm{j}\\4-6\mathrm{j}&5-5\mathrm{j}&6-4\mathrm{j}\\7-3\mathrm{j}&8-2\mathrm{j}&-\mathrm{j}\end{bmatrix}$$

2.5.2 矩阵的转置与旋转

在对矩阵进行处理时，有时需要用到矩阵转置，有的时候可能需要对矩阵进行翻转和旋转处理，这些基本操作在MATLAB下都有现成的处理函数，总结如下。

1) 矩阵转置

这里给出两类矩阵转置运算。

定义2-14　假设矩阵 $\boldsymbol{A}\in\mathscr{C}^{n\times m}$，则其转置矩阵 $\boldsymbol{B}$ 的元素定义为 $b_{ji}=a_{ij},\ i=1,2,\cdots,n,j=1,2,\cdots,m$，故 $\boldsymbol{B}\in\mathscr{C}^{m\times n}$ 矩阵，记作 $\boldsymbol{B}=\boldsymbol{A}^{\mathrm{T}}$，这种转置又称为直接转置。

定义2-15　如果矩阵 $\boldsymbol{A}$ 含有复数元素，则对之进行转置时，其转置矩阵 $\boldsymbol{B}$ 的元素定义为 $b_{ji}=a_{ij}^{\mathrm{H}},i=1,2,\cdots,n,j=1,2,\cdots,m$，亦即首先对各个元素进行转置，然后再逐项求取其共轭复数值。这种转置方式又称为Hermite转置，记作 $\boldsymbol{B}=\boldsymbol{A}^{\mathrm{H}}$。

MATLAB中用 $\boldsymbol{B}$=$\boldsymbol{A}$' 可以求出 $\boldsymbol{A}$ 矩阵的Hermite转置，矩阵的转置则可以由 $\boldsymbol{C}$=$\boldsymbol{A}$.' 求出。

定义2-16　若复数矩阵等于其共轭转置，即 $\boldsymbol{A}=\boldsymbol{A}^{\mathrm{H}}$，则 $\boldsymbol{A}$ 称为Hermite矩阵。

定义2-17　若复数矩阵 $\boldsymbol{A}=-\boldsymbol{A}^{\mathrm{H}}$，则 $\boldsymbol{A}$ 称反Hermite（skew-Hermitian）矩阵。

定理2-1　复数矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 乘积的转置满足

$$(\boldsymbol{AB})^{\mathrm{T}}=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}},\ \ (\boldsymbol{AB})^{\mathrm{H}}=\boldsymbol{B}^{\mathrm{H}}\boldsymbol{A}^{\mathrm{H}}\tag{2-5-1}$$

例2-26　考虑例2-25中的复数矩阵 $\boldsymbol{B}$，试求其直接转置与Hermite转置。

解　先将矩阵 $\boldsymbol{B}$ 输入计算机，则可以由下面命令得出两种转置。

```
>> B=[1+9i,2+8i,3+7j; 4+6j 5+5i,6+4i; 7+3i,8+2j 1i];
   B1=B', B2=B.' %两种不同的转置方式
```

得出的Hermite转置与直接转置分别为

$$\boldsymbol{B}_1=\begin{bmatrix}1-9\mathrm{j}&4-6\mathrm{j}&7-3\mathrm{j}\\2-8\mathrm{j}&5-5\mathrm{j}&8-2\mathrm{j}\\3-7\mathrm{j}&6-4\mathrm{j}&-1\mathrm{j}\end{bmatrix},\ \ \boldsymbol{B}_2=\begin{bmatrix}1+9\mathrm{j}&4+6\mathrm{j}&7+3\mathrm{j}\\2+8\mathrm{j}&5+5\mathrm{j}&8+2\mathrm{j}\\3+7\mathrm{j}&6+4\mathrm{j}&1\mathrm{j}\end{bmatrix}$$

2) 矩阵翻转

MATLAB提供了一些矩阵翻转处理的特殊命令。例如，矩阵的左右翻转函数 $\boldsymbol{B}$=fliplr($\boldsymbol{A}$)，将矩阵 $\boldsymbol{A}$ 进行左右翻转再赋给 $\boldsymbol{B}$，亦即 $b_{ij}=a_{i,n+1-j}$。从效果

上看，左右翻转函数等效于$\boldsymbol{B}$=$\boldsymbol{A}$(:,end:−1:1)。而$\boldsymbol{C}$=flipud($\boldsymbol{A}$)命令将矩阵$\boldsymbol{A}$进行上下翻转并将结果赋给$\boldsymbol{C}$，亦即$c_{ij}=a_{m+1-i,j}$，矩阵的上下翻转命令等效于$\boldsymbol{C}$=$\boldsymbol{A}$(end:−1:1,1)。

矩阵左右翻转的另一种方法是$\boldsymbol{B}$=$\boldsymbol{A}$(:,end:−1:1)，类似地还可以实现上下翻转，这样做的好处是可以实现矩阵行列的任意排序或局部矩阵排序。

例2-27　已知如下矩阵，试将其各行作一次随机排列。

$$\boldsymbol{A}=\begin{bmatrix}6&1&1&2&5\\6&3&3&5&0\\2&0&4&2&4\end{bmatrix}$$

解　利用randperm(n)函数可以实现$1,2,\cdots,n$的一次随机排序，利用该排序则可以实现$\boldsymbol{A}$矩阵各行的随机排序。

```
>> A=[6,1,1,2,5; 6,3,3,5,0; 2,0,4,2,4];
   ii=randperm(3), B=A(ii,:)
```

得出的随机排序次序向量为$[2,1,3]$，重新排序结果为(每次运行结果可能不同)

$$\boldsymbol{B}=\begin{bmatrix}6&3&3&5&0\\6&1&1&2&5\\2&0&4&2&4\end{bmatrix}$$

3) 矩阵的旋转

MATLAB函数$\boldsymbol{D}$=rot90($\boldsymbol{A}$)可以将$\boldsymbol{A}$矩阵逆时针旋转90°后赋给$\boldsymbol{D}$，亦即$d_{ij}=a_{m+1-i,j}$。函数$\boldsymbol{E}$=rot90($\boldsymbol{A}$,k)还可以逆时针旋转该矩阵$90k$°后赋给$\boldsymbol{E}$矩阵，其中k为整数。

例2-28　已知如下的$\boldsymbol{A}$矩阵，试将其顺时针旋转90°，转成$\boldsymbol{B}$矩阵的形式。

$$\boldsymbol{A}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}7&4&1\\8&5&2\\0&6&3\end{bmatrix}$$

解　标准的rot90()函数处理的是逆时针旋转的问题。矩阵顺时针旋转90°有两种方法实现，第一种在调用rot90()时令$k=-1$，另一种是令$k=3$，即逆时针旋转270°。所以，下面的语句可以直接得出旋转矩阵$\boldsymbol{B}_1=\boldsymbol{B}_2$，都是所需的$\boldsymbol{B}$矩阵。

```
>> A=[1 2 3; 4 5 6; 7 8 0]; B1=rot90(A,-1), B2=rot90(A,3)
```

2.5.3　矩阵的代数运算

代数运算是MATLAB科学运算领域很基础的一类运算。本节将给出代数运算的定义，然后介绍基于MATLAB的代数运算实现方法。

定义2-18　变量之间的有限次加、减、乘、除、乘方、开方等运算称为代数运算。

MATLAB语言中定义了下面各种矩阵的基本代数运算：

1) 加减法运算

假设在MATLAB工作空间下有两个矩阵$\boldsymbol{A}$和$\boldsymbol{B}$，则可以由$\boldsymbol{C}$=$\boldsymbol{A}$+$\boldsymbol{B}$和$\boldsymbol{C}=\boldsymbol{A}-\boldsymbol{B}$命令执行矩阵加减法。若$\boldsymbol{A}$、$\boldsymbol{B}$的维数相同，则自动地将$\boldsymbol{A}$、$\boldsymbol{B}$的相应元素相加减，从而得出正确的结果，并赋给$\boldsymbol{C}$变量。

MATLAB下考虑了两种特殊情况，允许不同维数的矩阵作加减运算。

（1）若二者之一为标量，则应该将其遍加（减）于另一个矩阵。

（2）若$\boldsymbol{A}\in\mathscr{C}^{n\times m}$，$\boldsymbol{B}$为$n\times 1$列向量或$1\times m$行向量，早期版本的MATLAB版本会给出错误信息，而新版本的MATLAB允许将列向量或行向量遍加或遍减到另一个矩阵的各列或各行上去，得出新的和矩阵或差矩阵。

在其他情况下，MATLAB将自动地给出错误信息，提示用户两个矩阵的维数不匹配。

例2-29　观察两个简单的变量如下，它们的和$\boldsymbol{A}+\boldsymbol{B}$是多少？

$$\boldsymbol{A}=\begin{bmatrix}5\\6\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}1&2\\3&4\end{bmatrix}$$

解　在数学上这两个矩阵是不可加的，早期版本的MATLAB如果作加法也将得到错误信息，在新发布的MATLAB下可以尝试下面的加减法运算。

```
>> A=[5;6]; B=[1 2; 3 4]; C=A+B, D=B-A'
```

实际应用中可以定义出一种有意义的“加法”：因为$\boldsymbol{A}$是列向量，所以将其遍加到$\boldsymbol{B}$矩阵的各列上，可以得出“加法矩阵”如下。另外，由于$\boldsymbol{A}^{\mathrm{T}}$为行向量，$\boldsymbol{D}$矩阵等于$\boldsymbol{B}$矩阵每行遍减$\boldsymbol{A}^{\mathrm{T}}$向量得出的矩阵。

$$\boldsymbol{C}=\begin{bmatrix}6&7\\9&10\end{bmatrix},\ \boldsymbol{D}=\begin{bmatrix}-4&-4\\-2&-2\end{bmatrix}$$

2) 矩阵乘法

与两个矩阵乘法相关的定义如下。

定义2-19　假设有两个矩阵$\boldsymbol{A}$和$\boldsymbol{B}$，其中，$\boldsymbol{A}$矩阵的列数与$\boldsymbol{B}$矩阵的行数相等，或其一为标量，则称$\boldsymbol{A}$，$\boldsymbol{B}$矩阵是可乘的，或称$\boldsymbol{A}$和$\boldsymbol{B}$矩阵的维数是相容的。

定义2-20　假设$\boldsymbol{A}\in\mathscr{C}^{n\times m}$，$\boldsymbol{B}\in\mathscr{C}^{m\times r}$，则$\boldsymbol{C}=\boldsymbol{AB}\in\mathscr{C}^{n\times r}$，满足

$$c_{ij}=\sum_{k=1}^{m}a_{ik}b_{kj},\quad i=1,2,\cdots,n,\ j=1,2,\cdots,r \tag{2-5-2}$$

定义2-21　如果两个矩阵$\boldsymbol{A}$与$\boldsymbol{B}$任何一个为标量，则$\boldsymbol{AB}$等于将这个标量遍乘到另一个矩阵每个元素后的新矩阵。

MATLAB语言中两个矩阵的乘法由$\boldsymbol{C}$=$\boldsymbol{A}$*$\boldsymbol{B}$直接求出，且这里并不需要指定$\boldsymbol{A}$和$\boldsymbol{B}$矩阵的维数。若$\boldsymbol{A}$和$\boldsymbol{B}$矩阵的维数相容，则可以准确无误地获得乘积矩阵$\boldsymbol{C}$；如果二者的维数不相容，则将给出错误信息，通知用户两个矩阵不可乘。

例2-30　已知两个矩阵$\boldsymbol{A}$与$\boldsymbol{B}$如下，由MATLAB乘法命令可以得出矩阵的积。

$$\boldsymbol{A}=\begin{bmatrix}5 & 1 & 2\\ 0 & 4 & 5\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}-1 & 0\\ -1 & -2\\ 0 & 3\end{bmatrix}$$

解　矩阵乘法是矩阵运算的基础，这里通过底层乘法的方式演示这两个矩阵相乘的结果。该结果与$\boldsymbol{A}*\boldsymbol{B}$命令的结果是完全一致的。

$$\boldsymbol{AB}=\begin{bmatrix}5\times(-1)+1\times(-1)+2\times 0 & 5\times 0+1\times(-2)+2\times 3\\ 0\times(-1)+4\times(-1)+5\times 0 & 0\times 0+4\times(-2)+5\times 3\end{bmatrix}=\begin{bmatrix}-6 & 4\\ -4 & 7\end{bmatrix}$$

在MATLAB下还可以尝试$\boldsymbol{A}*\boldsymbol{B}$'命令，不过该命令将导致错误信息，说明第一个矩阵的列数与第二个矩阵的行数不匹配，二者不能相乘。因为要执行这样的运算，$\boldsymbol{B}$'的计算优先于乘法运算，从而其转置为2×3矩阵，这样，$\boldsymbol{A}$乘以该矩阵由于维数不匹配而不能相乘，导致错误信息。

3) 向量的内积

向量的内积是一个行向量与一个列向量的乘积，结果为标量。

定义2-22　已知两个等长的列向量$\boldsymbol{a}$、$\boldsymbol{b}$，其内积定义为$\langle\boldsymbol{a},\boldsymbol{b}\rangle=\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}$。

定理2-2　内积满足下面一些性质。

$$\langle\boldsymbol{a},\boldsymbol{b}\rangle=\langle\boldsymbol{b},\boldsymbol{a}\rangle,\ \langle\lambda\boldsymbol{a},\boldsymbol{b}\rangle=\lambda\langle\boldsymbol{a},\boldsymbol{b}\rangle,\ \langle\boldsymbol{a},\boldsymbol{b}+\boldsymbol{c}\rangle=\langle\boldsymbol{a},\boldsymbol{b}\rangle+\langle\boldsymbol{a},\boldsymbol{c}\rangle \qquad (2\text{-}5\text{-}3)$$

定理2-3　当且仅当$\langle\boldsymbol{a},\boldsymbol{a}\rangle=0$时，$\boldsymbol{a}\equiv\boldsymbol{0}$。

如果MATLAB工作空间中有两个向量$\boldsymbol{a}$、$\boldsymbol{b}$，其内积可以由c=$\boldsymbol{a}$(:).'*$\boldsymbol{b}$(:)求出，即使这两个向量不是列向量也能直接求解。

4) 矩阵的左除

MATLAB中用“\”运算符号表示两个矩阵的左除，$\boldsymbol{A}\backslash\boldsymbol{B}$为方程$\boldsymbol{AX}=\boldsymbol{B}$的解$\boldsymbol{X}$。若$\boldsymbol{A}$为非奇异方阵，则$\boldsymbol{X}=\boldsymbol{A}^{-1}\boldsymbol{B}$。如果$\boldsymbol{A}$矩阵不是方阵，也可以求出$\boldsymbol{X}$=$\boldsymbol{A}\backslash\boldsymbol{B}$，这时将使用最小二乘解法来求取$\boldsymbol{AX}=\boldsymbol{B}$中的$\boldsymbol{X}$矩阵。

5) 矩阵的右除

MATLAB中定义了“/”符号，用于表示两个矩阵的右除，相当于求方程$\boldsymbol{XA}=\boldsymbol{B}$的解。$\boldsymbol{A}$为非奇异方阵时，$\boldsymbol{B}/\boldsymbol{A}$为$\boldsymbol{BA}^{-1}$，但在计算方法上存在差异，更精确地，有$\boldsymbol{B}/\boldsymbol{A}$=($\boldsymbol{A}$'\$\boldsymbol{B}$')'。

6) 矩阵乘方运算

一个矩阵的乘方运算可以在数学上表述成$\boldsymbol{A}^x$。如果x为正整数，则乘方表达式$\boldsymbol{A}^x$的结果可以将$\boldsymbol{A}$矩阵自乘x次得出。如果x为负整数，则可以将$\boldsymbol{A}$矩阵自乘x次，然后对结果进行求逆运算就可以得出该乘方结果。如果x是一个分数，例如

$x=n/m$，其中n和m均为整数，则相当于将$\boldsymbol{A}$矩阵自乘n次，然后对结果再开m次方。在MATLAB中统一表示成$\boldsymbol{F}=\boldsymbol{A}\hat{}x$。

7) 矩阵开方运算

数学公式上看，矩阵$\boldsymbol{A}$自乘n次是可以得出唯一解的，而其结果再作m次开方则应该有m个不同的根。考虑$\sqrt[3]{-1}$，其一个根是-1，对该根在复数平面内旋转$120°$可以得到第二个根，再旋转$120°$则可以得出第三个根。怎么实现旋转$120°$呢？可以将结果乘以复数标量$\delta=\mathrm{e}^{2\pi\mathrm{j}/3}$实现。

定理2-4　若$\boldsymbol{A}$矩阵的一个m次方根矩阵为$\boldsymbol{A}_0$，则其他m次方根为$\boldsymbol{A}_0\mathrm{e}^{2k\pi\mathrm{j}/m}$。其中，$k=1,2,\cdots,m-1$。

使用MATLAB通过的$\boldsymbol{A}$^(1/m)命令就可以得到矩阵的一个m次方根。

例2-31　考虑下面给出的$\boldsymbol{A}$矩阵，试求出其全部立方根并检验结果。

$$\boldsymbol{A}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix}$$

解　由乘方运算可以容易地得出原矩阵的一个立方根。

```
>> A=[1,2,3; 4,5,6; 7,8,0]; C=A^(1/3),
   e=norm(A-C^3) %求立方根并检验
```

具体表示如下。经检验，误差范数为$e=1.0145\times10^{-14}$，比较精确。

$$\boldsymbol{C}=\begin{bmatrix}0.77179+\mathrm{j}0.65380&0.48688-\mathrm{j}0.01592&0.17642-\mathrm{j}0.2887\\0.88854-\mathrm{j}0.07257&1.44730+\mathrm{j}0.47937&0.52327-\mathrm{j}0.4959\\0.46846-\mathrm{j}0.64647&0.66929-\mathrm{j}0.67480&1.33790+\mathrm{j}1.0488\end{bmatrix}$$

事实上，矩阵的立方根应该有三个结果，而上面只得出其中的一个。对该方根进行两次旋转，即计算$\boldsymbol{C}\mathrm{e}^{\mathrm{j}2\pi/3}$和$\boldsymbol{C}\mathrm{e}^{\mathrm{j}4\pi/3}$，则将得出另外两个根。

```
>> j1=exp(sqrt(-1)*2*pi/3);
   A1=C*j1, A2=C*j1^2                    %通过旋转求另外两个根
   e1=norm(A-A1^3), e2=norm(A-A2^3)      %矩阵方根的直接检验
```

这样可以得出另外两个根如下，误差都是10^{-14}级别的。

$$\boldsymbol{A}_1=\begin{bmatrix}-0.95210+\mathrm{j}0.34149&-0.22966+\mathrm{j}0.42961&0.16181+\mathrm{j}0.29713\\-0.38142+\mathrm{j}0.80579&-1.13880+\mathrm{j}1.01370&0.16784+\mathrm{j}0.70112\\0.32563+\mathrm{j}0.72893&0.24974+\mathrm{j}0.91702&-1.57720+\mathrm{j}0.63425\end{bmatrix}$$

$$\boldsymbol{A}_2=\begin{bmatrix}0.18031-\mathrm{j}0.99529&-0.25722-\mathrm{j}0.41369&-0.33823-\mathrm{j}0.00844\\-0.50712-\mathrm{j}0.73321&-0.30850-\mathrm{j}1.49310&-0.69111-\mathrm{j}0.20521\\-0.79409-\mathrm{j}0.08246&-0.91904-\mathrm{j}0.24222&0.23934-\mathrm{j}1.68310\end{bmatrix}$$

还可以考虑在符号运算的框架下由变精度算法计算已知矩阵的立方根，精度将达到7.2211×10^{-39}，精度远高于双精度框架下的计算结果。

```
>> A=sym([1,2,3; 4,5,6; 7,8,0]); C=A^(sym(1/3));
   C=vpa(C); norm(C^3-A) %高精度解
```

例2-32　矩阵$\boldsymbol{A}$的逆矩阵在数学上记作$\boldsymbol{A}^{-1}$,并可以由inv($\boldsymbol{A}$)函数直接计算。试求例2-25的复数矩阵的-1次方,看看是不是等于$\boldsymbol{B}$矩阵的逆矩阵。

解　为保证计算精度,下面的计算在符号运算框架下实现。

```
>> B=[1+9i,2+8i,3+7j; 4+6j 5+5i,6+4i; 7+3i,8+2j 1i];
   B=sym(B); B1=B^(-1), B2=inv(B), C=B1*B
```

由上面语句可以看出二者是相等的,都是正确的,且与原矩阵的乘积是单位矩阵,这也说明矩阵的逆确实是矩阵的“倒数”。

$$\boldsymbol{B}_1=\boldsymbol{B}_2=\begin{bmatrix}13/18-5\mathrm{j}/6 & -10/9+\mathrm{j}/3 & -1/9\\ -7/9+2\mathrm{j}/3 & 19/18-\mathrm{j}/6 & 2/9\\ -1/9 & 2/9 & -1/9\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$$

8) 点运算

MATLAB中定义了一种特殊的运算,即所谓的点运算。两个矩阵之间的点运算是它们对应元素的直接运算。例如,$\boldsymbol{C}$=$\boldsymbol{A}$.*$\boldsymbol{B}$表示$\boldsymbol{A}$和$\boldsymbol{B}$矩阵的相应元素之间直接进行乘法运算,然后将结果赋给$\boldsymbol{C}$矩阵,即$c_{ij}=a_{ij}b_{ij}$。这种点乘积运算又称为Hadamard乘积。注意,点乘积运算要求$\boldsymbol{A}$和$\boldsymbol{B}$矩阵的维数相同,或其一为标量。可以看出,这种运算和普通乘法运算是不同的。

点运算在MATLAB中起着很重要的作用。例如,当$\boldsymbol{x}$是一个向量时,则求取数值$[x_i^5]$时不能直接写成$\boldsymbol{x}$^5,而必须写成$\boldsymbol{x}$.^5。在进行矩阵的点运算时,同样要求运算的两个矩阵的维数一致,或其中一个变量为标量。其实一些特殊的函数,如sin()也是由点运算的形式进行的,因为它要对矩阵的每个元素求取正弦值。

矩阵点运算不只可以用于点乘积运算,还可以用于其他运算的场合。

例2-33　对例2-1给出的矩阵$\boldsymbol{A}$,试求解并理解$\boldsymbol{B}$=$\boldsymbol{A}$.^$\boldsymbol{A}$运算。

解　对前面给出的矩阵$\boldsymbol{A}$作$\boldsymbol{B}$=$\boldsymbol{A}$.^$\boldsymbol{A}$运算,则新矩阵的第(i,j)个元素为$b_{i,j}=a_{ij}^{a_{ij}}$,这样可以得出下面的结果。

```
>> A=[-1,5,4,6; 0,2,4,-2; 4,0,-2,5]; B=A.^A
   A=sym(A); C=A.^A %对应元素单独运算可以求点乘方
```

该语句将计算并生成如下的矩阵的数值解与解析解,分别为

$$\boldsymbol{B}=\begin{bmatrix}-1&3125&256&46656\\1&4&256&0.25\\256&1&0.25&3125\end{bmatrix},\ \boldsymbol{C}=\begin{bmatrix}-1&3125&256&46656\\1&4&256&1/4\\256&1&1/4&3125\end{bmatrix}$$

从结果看,与预期一致,得出的结果是由下面矩阵得出的。

$$\boldsymbol{A}.^{\wedge}\boldsymbol{A}=\begin{bmatrix}(-1)^{(-1)}&5^5&4^4&6^6\\0^0&2^2&4^4&(-2)^{(-2)}\\4^4&0^0&(-2)^{(-2)}&5^5\end{bmatrix}$$

2.5.4　矩阵的Kronecker乘积与Kronecker和

Kronecker乘积与Kronecker和是以德国数学家、逻辑学家Leopold Kronecker（1823−1891）命名的运算，在线性方程求解中有着重要的应用。

定义2-23　两个矩阵$\boldsymbol{A}$与$\boldsymbol{B}$，其Kronecker乘积定义为

$$\boldsymbol{C}=\boldsymbol{A}\otimes\boldsymbol{B}=\begin{bmatrix}a_{11}\boldsymbol{B} & \cdots & a_{1m}\boldsymbol{B}\\ \vdots & \ddots & \vdots\\ a_{n1}\boldsymbol{B} & \cdots & a_{nm}\boldsymbol{B}\end{bmatrix} \tag{2-5-4}$$

定义2-24　矩阵$\boldsymbol{A}$与$\boldsymbol{B}$的Kronecker和$\boldsymbol{A}\oplus\boldsymbol{B}$的数学定义为

$$\boldsymbol{D}=\boldsymbol{A}\oplus\boldsymbol{B}=\begin{bmatrix}a_{11}+\boldsymbol{B} & \cdots & a_{1m}+\boldsymbol{B}\\ \vdots & \ddots & \vdots\\ a_{n1}+\boldsymbol{B} & \cdots & a_{nm}+\boldsymbol{B}\end{bmatrix} \tag{2-5-5}$$

定理2-5　若矩阵$\boldsymbol{A}$与$\boldsymbol{B}$维数相同，则Kronecker乘积满足分配律

$$\begin{aligned}(\boldsymbol{A}+\boldsymbol{B})\otimes\boldsymbol{C}&=\boldsymbol{A}\otimes\boldsymbol{C}+\boldsymbol{B}\otimes\boldsymbol{C}\\ \boldsymbol{C}\otimes(\boldsymbol{A}+\boldsymbol{B})&=\boldsymbol{C}\otimes\boldsymbol{A}+\boldsymbol{C}\otimes\boldsymbol{B}\end{aligned} \tag{2-5-6}$$

定理2-6　Kronecker乘积的转置为

$$\big(\boldsymbol{A}\otimes\boldsymbol{B}\big)^{\mathrm{T}}=\boldsymbol{B}^{\mathrm{T}}\otimes\boldsymbol{A}^{\mathrm{T}} \tag{2-5-7}$$

定理2-7　Kronecker乘积的结合律为

$$\boldsymbol{A}\big(\boldsymbol{B}\otimes\boldsymbol{C}\big)=\big(\boldsymbol{A}\otimes\boldsymbol{B}\big)\boldsymbol{C} \tag{2-5-8}$$

定理2-8　如果下面参与乘积的矩阵维数相容，则

$$\big(\boldsymbol{A}\otimes\boldsymbol{B}\big)\big(\boldsymbol{C}\otimes\boldsymbol{D}\big)=(\boldsymbol{AC})\otimes(\boldsymbol{BD}) \tag{2-5-9}$$

上面几个定理中，如果$\otimes$运算符替换成$\oplus$，定理仍然成立。

与矩阵的常规加法和乘法不同，Kronecker和与乘积并不要求两个矩阵有相容性。另外，Kronecker和与乘积不满足交换律。

MATLAB中提供的函数$\boldsymbol{C}$=`kron(`$\boldsymbol{A}$`,`$\boldsymbol{B}$`)`可直接计算两个矩阵的Kronecker积$\boldsymbol{A}\otimes\boldsymbol{B}$。仿照该函数，可以编写出Kronecker和的求解函数`kronsum()`。

```
function C=kronsum(A,B)
[ma,na]=size(A); [mb,nb]=size(B);
A=reshape(A,[1 ma 1 na]); B=reshape(B,[mb 1 nb 1]);
C=reshape(bsxfun(@plus,A,B),[ma*mb na*nb]);
```

例2-34　已知如下两个矩阵$\boldsymbol{A}$与$\boldsymbol{B}$，试求其Kronecker乘积与Kronecker和。

$$\boldsymbol{A}=\begin{bmatrix}-2 & 2\\ 0 & -1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}-2 & -1 & 1\\ 0 & 1 & -2\end{bmatrix}$$

解　可以先输入这两个矩阵，然后调用`kron()`函数，分别计算出$\boldsymbol{A}\otimes\boldsymbol{B}$与$\boldsymbol{B}\otimes\boldsymbol{A}$。

```
>> A=[-2,2; 0,-1]; B=[-2,-1,1; 0,1,-2];
   kron(A,B), kron(B,A)
```

得出的结果如下所示。可见二者不同，说明 Kronecker 乘积不满足交换律。

$$\boldsymbol{A}\otimes\boldsymbol{B}=\left[\begin{array}{cc:cc:cc}4&2&-2&-4&-2&2\\0&-2&4&0&2&-4\\\hdashline 0&0&0&2&1&-1\\0&0&0&0&-1&2\end{array}\right],\ \boldsymbol{B}\otimes\boldsymbol{A}=\left[\begin{array}{cc:cc:cc}4&-4&2&-2&-2&2\\0&2&0&1&0&-1\\\hdashline 0&0&-2&2&4&-4\\0&0&0&-1&0&2\end{array}\right]$$

还可以由下面的语句计算 $\boldsymbol{A}\oplus\boldsymbol{B}$ 与 $\boldsymbol{B}\oplus\boldsymbol{A}$。

```
>> kronsum(A,B), kronsum(B,A)
```

得出的结果如下所示，说明 Kronecker 和也不满足交换律。

$$\boldsymbol{A}\oplus\boldsymbol{B}=\left[\begin{array}{ccc:ccc}-4&-3&-1&0&1&3\\-2&-1&-4&2&3&0\\\hdashline -2&-1&1&-3&-2&0\\0&1&-2&-1&0&-3\end{array}\right],\ \boldsymbol{B}\oplus\boldsymbol{A}=\left[\begin{array}{ccc:ccc}-4&0&-3&1&-1&3\\-2&-3&-1&-2&1&0\\\hdashline -2&2&-1&3&-4&0\\0&-1&1&0&-2&-3\end{array}\right]$$

2.6　矩阵函数的微积分运算

本节首先给出矩阵函数的导数和积分定义。在此基础上，还将介绍各种复杂函数矩阵的导函数矩阵计算方法与应用，以及 Jacobi 矩阵与 Hesse 矩阵的概念与计算问题。

2.6.1　矩阵函数的导数

定义 2-25　若矩阵 $\boldsymbol{A}(t)$ 是 t 的函数，则 $\boldsymbol{A}(t)$ 对 t 的导数等于每个元素单独求导的矩阵，即

$$\frac{\mathrm{d}\boldsymbol{A}(t)}{\mathrm{d}t}=\left[\frac{\mathrm{d}}{\mathrm{d}t}a_{ij}(t)\right]\tag{2-6-1}$$

如果已知矩阵 $\boldsymbol{A}(t)$ 是 t 的函数，则 $\mathrm{d}\boldsymbol{A}(t)/\mathrm{d}t$ 可以由 `diff()` 函数直接求出。

例 2-35　试求出下面函数矩阵的导数。

$$\boldsymbol{A}(t)=\left[\begin{array}{ccc}t^2/2+1&t&t^2/2\\t&1&t\\-t^2/2&-t&1-t^2/2\end{array}\right]\mathrm{e}^{-t}$$

解　可以先将矩阵函数输入 MATLAB 环境，然后直接对其求导。由于这里使用了矩阵函数的表示方法，为避免与已有矩阵冲突，应该用 `clear` 命令先清除原有的 $\boldsymbol{A}$ 矩阵，再输入新矩阵。

```
>> syms t; clear A
   A(t)=[t^2/2+1,t,t^2/2; t,1,t; -t^2/2,-t,1-t^2/2]*exp(-t);
   A1=simplify(diff(A,t))
```

得出的导函数矩阵为

$$\boldsymbol{A}_1(t)=\left[\begin{array}{ccc}-t^2/2+t-1&1-t&-t(t-2)/2\\1-t&-1&1-t\\t(t-2)/2&t-1&t^2/2-t-1\end{array}\right]\mathrm{e}^{-t}$$

定理2-9　假设$\boldsymbol{A}(t)$和$\boldsymbol{B}(t)$都可微，则

$$\frac{\mathrm{d}}{\mathrm{d}t}\big[\boldsymbol{A}(t)+\boldsymbol{B}(t)\big]=\frac{\mathrm{d}\boldsymbol{A}(t)}{\mathrm{d}t}+\frac{\mathrm{d}\boldsymbol{B}(t)}{\mathrm{d}t} \tag{2-6-2}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\big[\boldsymbol{A}(t)\boldsymbol{B}(t)\big]=\boldsymbol{A}(t)\frac{\mathrm{d}\boldsymbol{B}(t)}{\mathrm{d}t}+\frac{\mathrm{d}\boldsymbol{A}(t)}{\mathrm{d}t}\boldsymbol{B}(t) \tag{2-6-3}$$

例2-36　考虑例2-35中的$\boldsymbol{A}(t)$与如下$\boldsymbol{B}(t)$矩阵，试求$\boldsymbol{A}(t)\boldsymbol{B}(t)$对$t$的导数。

$$\boldsymbol{B}(t)=\begin{bmatrix}\mathrm{e}^{-t}-\mathrm{e}^{-2t}+\mathrm{e}^{-3t} & \mathrm{e}^{-t}-\mathrm{e}^{-2t} & \mathrm{e}^{-t}-\mathrm{e}^{-3t}\\ 2\mathrm{e}^{-2t}-\mathrm{e}^{-t}-\mathrm{e}^{-3t} & 2\mathrm{e}^{-2t}-\mathrm{e}^{-t} & \mathrm{e}^{-3t}-\mathrm{e}^{-t}\\ \mathrm{e}^{-t}-\mathrm{e}^{-2t} & \mathrm{e}^{-t}-\mathrm{e}^{-2t} & \mathrm{e}^{-t}\end{bmatrix}$$

解　可以用两种方法求$\boldsymbol{A}(t)\boldsymbol{B}(t)$的导数，一种方法是先求出$\boldsymbol{A}(t)\boldsymbol{B}(t)$再用diff()函数直接求导；另一种方法是由式(2-6-3)求导，两种方法得出的结果完全一致。从这个例子看，对复杂函数求导问题没有必要借助于式(2-6-3)介绍的间接方法，直接调用diff()函数求解即可。由于得出的结果过于繁杂，这里就不列出了。

```
>> syms t; clear A B
   A(t)=[t^2/2+1,t,t^2/2; t,1,t; -t^2/2,-t,1-t^2/2]*exp(-t);
   B(t)=[exp(-t)-exp(-2*t)+exp(-3*t),...
              exp(-t)-exp(-2*t),exp(-t)-exp(-3*t);
         2*exp(-2*t)-exp(-t)-exp(-3*t),...
              2*exp(-2*t)-exp(-t),exp(-3*t)-exp(-t);
         exp(-t)-exp(-2*t), exp(-t)-exp(-2*t), exp(-t)];
   A1=simplify(A*diff(B)+diff(A)*B)
   A2=simplify(diff(A*B)), simplify(A1-A2)
```

2.6.2　矩阵函数的积分

定义2-26　矩阵函数$\boldsymbol{A}(t)$对t的定积分定义为每个元素定积分构成的矩阵。

$$\int_{t_0}^{t_\mathrm{n}}\boldsymbol{A}(t)\mathrm{d}t=\left\{\int_{t_0}^{t_\mathrm{n}}a_{ij}(t)\mathrm{d}t\right\} \tag{2-6-4}$$

MATLAB提供的int()函数可以直接对矩阵进行不定积分、定积分或反常积分运算。下面将通过例子演示积分运算。

例2-37　试对例2-35的结果进行积分运算，验证能否返回原函数。

解　可以先输入$\boldsymbol{A}(t)$函数矩阵，对其求导再对结果作不定积分运算。可以看出，最终得出的结果可以还原给定的矩阵。

```
>> syms t; clear A
   A(t)=[t^2/2+1,t,t^2/2; t,1,t; -t^2/2,-t,1-t^2/2]*exp(-t);
   A1=simplify(diff(A,t)); A2=simplify(int(A1))
```

2.6.3　向量函数的 Jacobi 矩阵

定义 2-27　假设有 n 个自变量的 m 个函数定义为

$$\begin{cases} y_1 = f_1(x_1, x_2, \cdots, x_n) \\ y_2 = f_2(x_1, x_2, \cdots, x_n) \\ \quad \vdots \\ y_m = f_m(x_1, x_2, \cdots, x_n) \end{cases} \tag{2-6-5}$$

将相应的 y_i 对 x_j 求偏导，则得出如下的 Jacobi 矩阵。

$$\boldsymbol{J} = \begin{bmatrix} \partial y_1/\partial x_1 & \partial y_1/\partial x_2 & \cdots & \partial y_1/\partial x_n \\ \partial y_2/\partial x_1 & \partial y_2/\partial x_2 & \cdots & \partial y_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial y_m/\partial x_1 & \partial y_m/\partial x_2 & \cdots & \partial y_m/\partial x_n \end{bmatrix} \tag{2-6-6}$$

Jacobi 矩阵（Jacobian matrix）是以德国数学家 Carl Gustav Jacob Jacobi（1804−1851）命名的，又称为梯度矩阵。Jacobi 矩阵可以由 MATLAB 的符号运算工具箱中的 `jacobian()` 函数直接求得，其调用格式为 $\boldsymbol{J}$=`jacobian(`$\boldsymbol{y}$`,`$\boldsymbol{x}$`)`。其中，$\boldsymbol{x}$ 是自变量构成的向量，$\boldsymbol{y}$ 是由各个函数构成的向量。

例 2-38　已知球面坐标到直角坐标的变换公式为 $x = r\sin\theta\cos\phi$，$y = r\sin\theta\sin\phi$，$z = r\cos\theta$，试求出函数向量 $[x, y, z]$ 对自变量向量 $[r, \theta, \phi]$ 的 Jacobi 矩阵。

解　先声明符号变量并描述三个函数，这样可以用下面语句求解出其 Jacobi 矩阵。

```
>> syms r theta phi; %声明符号变量
   x=r*sin(theta)*cos(phi); y=r*sin(theta)*sin(phi); z=r*cos(theta);
   J=jacobian([x; y; z],[r theta phi]) %直接求 Jacobi 矩阵
```

可以得出 Jacobi 矩阵为

$$\boldsymbol{J} = \begin{bmatrix} \sin\theta\cos\phi & r\cos\theta\cos\phi & -r\sin\theta\sin\phi \\ \sin\theta\sin\phi & r\cos\theta\sin\phi & r\sin\theta\cos\phi \\ \cos\theta & -r\sin\theta & 0 \end{bmatrix}$$

2.6.4　Hesse 矩阵

定义 2-28　给定 n 元标量函数 $f(x_1, x_2, \cdots, x_n)$ 的 Hesse 矩阵定义为

$$\boldsymbol{H} = \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1\partial x_2 & \cdots & \partial^2 f/\partial x_1\partial x_n \\ \partial^2 f/\partial x_2\partial x_1 & \partial^2 f/\partial x_2^2 & \cdots & \partial^2 f/\partial x_2\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial^2 f/\partial x_n\partial x_1 & \partial^2 f/\partial x_n\partial x_2 & \cdots & \partial^2 f/\partial x_n^2 \end{bmatrix} \tag{2-6-7}$$

Hesse 矩阵是以德国数学家 Ludwig Otto Hesse（1811−1874）命名的，该矩阵是标量函数 $f(x_1, x_2, \cdots, x_n)$ 的二阶偏导数矩阵。MATLAB 提供了 `hessian()` 函数，可以直接求出原函数的 Hesse 矩阵，调用格式为 $\boldsymbol{H}$=`hessian(`f`,`$\boldsymbol{x}$`)`。其中，向量

$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]$。

早期版本的MATLAB符号运算工具箱未提供`hessian()`函数，可以由嵌套调用的`jacobian()`函数求解，$\boldsymbol{H}$=`jacobian(jacobian(`f`,`$\boldsymbol{x}$`),`$\boldsymbol{x}$`)`。

例2-39　试求二元函数$f(x,y)=(x^2-2x)\mathrm{e}^{-x^2-y^2-xy}$的Hesse矩阵。

解　下面语句可以直接求取该函数的Hesse矩阵。

```
>> syms x y; f=(x^2-2*x)*exp(-x^2-y^2-x*y);
   H=simplify(hessian(f,[x,y]))
   H1=simplify(hessian(f,[x,y])/exp(-x^2-y^2-x*y))
```

提取指数再化简，得出的结果（或早期版本嵌套调用`jacobian()`函数）为

$$\boldsymbol{H}_1=\mathrm{e}^{-x^2-y^2-xy}\begin{bmatrix}4x-2(2x-2)(2x+y)-2x^2-(2x-x^2)(2x+y)^2+2 & 2x-(2x-2)(x+2y)-x^2-(2x-x^2)(x+2y)(2x+y)\\ 2x-(2x-2)(x+2y)-x^2-(2x-x^2)(x+2y)(2x+y) & x(x-2)(x^2+4xy+4y^2-2)\end{bmatrix}$$

本章习题

2.1 用MATLAB语句输入矩阵$\boldsymbol{A}$和$\boldsymbol{B}$。

$$\boldsymbol{A}=\begin{bmatrix}1&2&3&4\\4&3&2&1\\2&3&4&1\\3&2&4&1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1+4\mathrm{j}&2+3\mathrm{j}&3+2\mathrm{j}&4+1\mathrm{j}\\4+1\mathrm{j}&3+2\mathrm{j}&2+3\mathrm{j}&1+4\mathrm{j}\\2+3\mathrm{j}&3+2\mathrm{j}&4+1\mathrm{j}&1+4\mathrm{j}\\3+2\mathrm{j}&2+3\mathrm{j}&4+1\mathrm{j}&1+4\mathrm{j}\end{bmatrix}$$

$\boldsymbol{A}$为4×4矩阵，如果给出$\boldsymbol{A}(5,6)=5$命令，将得出什么结果？

2.2 试生成一个对角元素为$a_1,a_2,\cdots,a_{12}$的对角矩阵。

2.3 试从矩阵显示的形式辨认出矩阵是双精度矩阵还是符号矩阵。如果$\boldsymbol{A}$是数值矩阵而$\boldsymbol{B}$为符号矩阵，它们的乘积$\boldsymbol{C}$=$\boldsymbol{A}$*$\boldsymbol{B}$会是什么样的数据结构？试通过简单例子验证此判断。

2.4 试生成30000个标准正态分布的伪随机数，由得出的数据得出其均值与方差，并绘制其分布的直方图。

2.5 Jordan矩阵是矩阵分析中一类很实用的矩阵，其一般形式为

$$\boldsymbol{J}=\begin{bmatrix}-\alpha&1&0&\cdots&0\\0&-\alpha&1&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&-\alpha\end{bmatrix}$$

例如

$$\boldsymbol{J}_1=\begin{bmatrix}-5&1&0&0&0\\0&-5&1&0&0\\0&0&-5&1&0\\0&0&0&-5&1\\0&0&0&0&-5\end{bmatrix}$$

试利用`diag()`函数给出构造$\boldsymbol{J}_1$的语句。

2.6 已知$\boldsymbol{c}$=[−4,−3,−2,−1,0,1,2,3,4]，试由其生成Hankel矩阵、Vandermonde矩阵以及相伴矩阵。

2.7 试用底层命令编写出生成n阶Wilkinson矩阵的MATLAB函数。

2.8 试生成9×9的魔方矩阵，并观察数字$1\to 2\to\cdots\to 80\to 81$的走行规则。

2.9 试用随机矩阵的生成方式生成一个15×15矩阵，使得该矩阵的元素只有0和1，且矩阵行列式的值为1。

2.10 试不用循环方式将下面20×20矩阵输入计算机。

$$\boldsymbol{A}=\begin{bmatrix} x & a & a & \cdots & a\\ a & x & a & \cdots & a\\ a & a & x & \cdots & a\\ \vdots & \vdots & \vdots & \ddots & a\\ a & a & a & \cdots & x \end{bmatrix}$$

2.11 幂零矩阵是一类特殊的矩阵，其基本形式如下：矩阵的次主对角线元素为1，其余均为0。试验证：对指定阶次的幂零矩阵，有$\boldsymbol{H}_n^i=\boldsymbol{0}$对所有的$i\geqslant n$成立。

2.12 请将下面给出的矩阵$\boldsymbol{A}$和$\boldsymbol{B}$输入MATLAB环境，并将其转换成符号矩阵。

$$\boldsymbol{A}=\begin{bmatrix} 5 & 7 & 6 & 5 & 1 & 6 & 5\\ 2 & 3 & 1 & 0 & 0 & 1 & 4\\ 6 & 4 & 2 & 0 & 6 & 4 & 4\\ 3 & 9 & 6 & 3 & 6 & 6 & 2\\ 10 & 7 & 6 & 0 & 0 & 7 & 7\\ 7 & 2 & 4 & 4 & 0 & 7 & 7\\ 4 & 8 & 6 & 7 & 2 & 1 & 7 \end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix} 3 & 5 & 5 & 0 & 1 & 2 & 3\\ 3 & 2 & 5 & 4 & 6 & 2 & 5\\ 1 & 2 & 1 & 1 & 3 & 4 & 6\\ 3 & 5 & 1 & 5 & 2 & 1 & 2\\ 4 & 1 & 0 & 1 & 2 & 0 & 1\\ -3 & -4 & -7 & 3 & 7 & 8 & 12\\ 1 & -10 & 7 & -6 & 8 & 1 & 5 \end{bmatrix}$$

2.13 试对给出的$\boldsymbol{A}$矩阵求出$\boldsymbol{A}^{\mathrm{T}}$与$\boldsymbol{A}^{\mathrm{H}}$，并理解得出的结果。

$$\boldsymbol{A}=\begin{bmatrix} 4+3\mathrm{j} & 6+5\mathrm{j} & 3+3\mathrm{j} & 1+2\mathrm{j} & 4+\mathrm{j}\\ 6+4\mathrm{j} & 5+5\mathrm{j} & 1+2\mathrm{j} & 2\mathrm{j} & 5\\ 1 & 4 & 5+3\mathrm{j} & 5+\mathrm{j} & 4+6\mathrm{j} \end{bmatrix}$$

2.14 试生成5×5的魔方矩阵，并求出其所有的五次方根矩阵。

2.15 对3×3的魔方矩阵$\boldsymbol{A}$，试理解命令$\boldsymbol{A}$.^2与$\boldsymbol{A}$^2是否一致的，它们的物理意义是什么？

2.16 试生成一个9×9的魔方矩阵，并对其2~7列的顺序作一次随机排列，其他各列不变，试通过语句观察是不是能实现预期的结果。

2.17 已知下面给出的方阵$\boldsymbol{A}$，试用矩阵乘方的方式得出$\boldsymbol{A}$的逆矩阵，并求出$\boldsymbol{A}^{-5}$。

$$\boldsymbol{A}=\begin{bmatrix} 1 & 0 & 6 & 1\\ 3 & 6 & 1 & 2\\ 5 & 5 & 5 & 1\\ 5 & 1 & 6 & 3 \end{bmatrix}$$

2.18 对下面的矩阵$\boldsymbol{A}$和$\boldsymbol{B}$，试计算$\boldsymbol{A}\otimes\boldsymbol{B}$和$\boldsymbol{B}\otimes\boldsymbol{A}$，并判定二者是否相等。

$$\boldsymbol{A}=\begin{bmatrix} -1 & 2 & 2 & 1\\ -1 & 2 & 1 & 0\\ 2 & 1 & 1 & 0\\ 1 & 0 & 2 & 0 \end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix} 3 & 0 & 3\\ 3 & 2 & 2\\ 3 & 1 & 1 \end{bmatrix}$$

2.19 对习题2.18给出的矩阵，试求 $\boldsymbol{A}\oplus\boldsymbol{B}$ 与 $\boldsymbol{B}\oplus\boldsymbol{A}$，并判定二者是否相等。

2.20 已知如下矩阵，试验证 $\boldsymbol{A}_1\otimes\boldsymbol{B}_1+\boldsymbol{A}_2\otimes\boldsymbol{B}_2\neq(\boldsymbol{A}_1+\boldsymbol{A}_2)\otimes(\boldsymbol{B}_1+\boldsymbol{B}_2)$

$$\boldsymbol{A}_1=\begin{bmatrix}2&0\\2&2\end{bmatrix},\ \boldsymbol{A}_2=\begin{bmatrix}1&0\\0&1\end{bmatrix}$$

$$\boldsymbol{B}_1=\begin{bmatrix}2&2&1&1\\2&1&2&0\\0&2&2&2\\2&0&2&2\end{bmatrix},\ \boldsymbol{B}_2=\begin{bmatrix}2&1&0&2\\2&0&0&0\\2&2&0&2\\1&0&2&0\end{bmatrix}$$

2.21 已知函数矩阵 $\boldsymbol{A}(t)$、$\boldsymbol{B}(t)$，试求 $\mathrm{d}\big[\boldsymbol{A}^2(t)\boldsymbol{B}^3(t)\big]/\mathrm{d}t$。对结果求取积分计算，看看能否还原原来的矩阵。

$$\boldsymbol{A}(t)=\begin{bmatrix}t^2/2+1&t&t^2/2\\t&1&t\\-t^2/2&-t&1-t^2/2\end{bmatrix}\mathrm{e}^{-t}$$

$$\boldsymbol{B}(t)=\begin{bmatrix}\mathrm{e}^{-t}-\mathrm{e}^{-2t}+\mathrm{e}^{-3t}&\mathrm{e}^{-t}-\mathrm{e}^{-2t}&\mathrm{e}^{-t}-\mathrm{e}^{-3t}\\2\mathrm{e}^{-2t}-\mathrm{e}^{-t}-\mathrm{e}^{-3t}&2\mathrm{e}^{-2t}-\mathrm{e}^{-t}&\mathrm{e}^{-3t}-\mathrm{e}^{-t}\\\mathrm{e}^{-t}-\mathrm{e}^{-2t}&\mathrm{e}^{-t}-\mathrm{e}^{-2t}&\mathrm{e}^{-t}\end{bmatrix}$$

2.22 假设已知函数矩阵，试求出其Jacobi矩阵。

$$\boldsymbol{f}(x,y,z)=\begin{bmatrix}3x+\mathrm{e}^{y}z\\x^3+y^2\sin z\end{bmatrix}$$

2.23 已知三元标量函数 $f(x,y,z)=3x+\mathrm{e}^{y}z+x^3+y^2\sin z$，试求其Hesse矩阵。